1.14. /proc

/proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc). For this reason it can be regarded as a control and information centre for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory. For example, 'lsmod' is the same as 'cat /proc/modules' while 'lspci' is a synonym for 'cat /proc/pci'. By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

The most distinctive thing about files in this directory is the fact that all of them have a file size of 0, with the exception of kcore, mtrr and self. A directory listing looks similar to the following:

```
total 525256
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 1
dr-xr-xr-x    3 daemon   root            0 Jan 19 15:00 109
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 170
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 173
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 178
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 2
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 3
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 4
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 421
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 425
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 433
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 439
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 444
dr-xr-xr-x    3 daemon   daemon          0 Jan 19 15:00 446
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 449
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 453
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 456
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 458
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 462
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 463
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 464
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 465
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 466
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 467
dr-xr-xr-x    3 gdm      gdm             0 Jan 19 15:00 472
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 483
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 5
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 6
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 7
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 8
-r--r--r--    1 root     root            0 Jan 19 15:00 apm
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 bus
-r--r--r--    1 root     root            0 Jan 19 15:00 cmdline
-r--r--r--    1 root     root            0 Jan 19 15:00 cpuinfo
-r--r--r--    1 root     root            0 Jan 19 15:00 devices
-r--r--r--    1 root     root            0 Jan 19 15:00 dma
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 driver
-r--r--r--    1 root     root            0 Jan 19 15:00 execdomains
-r--r--r--    1 root     root            0 Jan 19 15:00 fb
-r--r--r--    1 root     root            0 Jan 19 15:00 filesystems
dr-xr-xr-x    2 root     root            0 Jan 19 15:00 fs
dr-xr-xr-x    4 root     root            0 Jan 19 15:00 ide
-r--r--r--    1 root     root            0 Jan 19 15:00 interrupts
```

```
-r--r--r--    1 root     root            0 Jan 19 15:00 iomem
-r--r--r--    1 root     root            0 Jan 19 15:00 ioports
dr-xr-xr-x   18 root     root            0 Jan 19 15:00 irq
-r--------    1 root     root     536809472 Jan 19 15:00 kcore
-r--------    1 root     root            0 Jan 19 14:58 kmsg
-r--r--r--    1 root     root            0 Jan 19 15:00 ksyms
-r--r--r--    1 root     root            0 Jan 19 15:00 loadavg
-r--r--r--    1 root     root            0 Jan 19 15:00 locks
-r--r--r--    1 root     root            0 Jan 19 15:00 mdstat
-r--r--r--    1 root     root            0 Jan 19 15:00 meminfo
-r--r--r--    1 root     root            0 Jan 19 15:00 misc
-r--r--r--    1 root     root            0 Jan 19 15:00 modules
-r--r--r--    1 root     root            0 Jan 19 15:00 mounts
-rw-r--r--    1 root     root          137 Jan 19 14:59 mtrr
dr-xr-xr-x    3 root     root            0 Jan 19 15:00 net
dr-xr-xr-x    2 root     root            0 Jan 19 15:00 nv
-r--r--r--    1 root     root            0 Jan 19 15:00 partitions
-r--r--r--    1 root     root            0 Jan 19 15:00 pci
dr-xr-xr-x    4 root     root            0 Jan 19 15:00 scsi
lrwxrwxrwx    1 root     root           64 Jan 19 14:58 self -> 483
-rw-r--r--    1 root     root            0 Jan 19 15:00 slabinfo
-r--r--r--    1 root     root            0 Jan 19 15:00 stat
-r--r--r--    1 root     root            0 Jan 19 15:00 swaps
dr-xr-xr-x   10 root     root            0 Jan 19 15:00 sys
dr-xr-xr-x    2 root     root            0 Jan 19 15:00 sysvipc
dr-xr-xr-x    4 root     root            0 Jan 19 15:00 tty
-r--r--r--    1 root     root            0 Jan 19 15:00 uptime
-r--r--r--    1 root     root            0 Jan 19 15:00 version
```

Each of the numbered directories corresponds to an actual process ID. Looking at the process table, you can match processes with the associated process ID. For example, the process table might indicate the following for the secure shell server:

```
# ps ax | grep sshd
439 ? S 0:00 /usr/sbin/sshd
```

Details of this process can be obtained by looking at the associated files in the directory for this process, /proc/460. You might wonder how you can see details of a process that has a file size of 0. It makes more sense if you think of it as a window into the kernel. The file doesn't actually contain any data; it just acts as a pointer to where the actual process information resides. For example, a listing of the files in the /proc/460 directory looks similar to the following:

```
total 0
-r--r--r--    1 root     root            0 Jan 19 15:02 cmdline
lrwxrwxrwx    1 root     root            0 Jan 19 15:02 cwd -> /
-r--------    1 root     root            0 Jan 19 15:02 environ
lrwxrwxrwx    1 root     root            0 Jan 19 15:02 exe -> /usr/sbin/sshd
dr-x------    2 root     root            0 Jan 19 15:02 fd
-r--r--r--    1 root     root            0 Jan 19 15:02 maps
-rw-------    1 root     root            0 Jan 19 15:02 mem
lrwxrwxrwx    1 root     root            0 Jan 19 15:02 root -> /
-r--r--r--    1 root     root            0 Jan 19 15:02 stat
-r--r--r--    1 root     root            0 Jan 19 15:02 statm
-r--r--r--    1 root     root            0 Jan 19 15:02 status
```

The purpose and contents of each of these files is explained below:

/proc/PID/cmdline

Command line arguments.

/proc/PID/cpu

Current and last cpu in which it was executed.

/proc/PID/cwd

Link to the current working directory.

/proc/PID/environ

Values of environment variables.

/proc/PID/exe

Link to the executable of this process.

/proc/PID/fd

Directory, which contains all file descriptors.

/proc/PID/maps

Memory maps to executables and library files.

/proc/PID/mem

Memory held by this process.

/proc/PID/root

Link to the root directory of this process.

/proc/PID/stat

Process status.

/proc/PID/statm

Process memory status information.

/proc/PID/status

Process status in human readable form.

Should you wish to know more, the man page for proc describes each of the files associated with a running process ID in far greater detail.

Even though files appear to be of size 0, examining their contents reveals otherwise:

```
# cat status

Name: sshd
State: S (sleeping)
Tgid: 439
Pid: 439
PPid: 1
TracerPid: 0
Uid: 0 0 0 0
```

```
Gid: 0 0 0 0
FDSize: 32
Groups:
VmSize:     2788 kB
VmLck:         0 kB
VmRSS:      1280 kB
VmData:      252 kB
VmStk:        16 kB
VmExe:       268 kB
VmLib:      2132 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 8000000000001000
SigCgt: 0000000000014005
CapInh: 0000000000000000
CapPrm: 00000000fffffeff
CapEff: 00000000fffffeff
```

The files in the /proc directory act very similar to the process ID subdirectory files. For example, examining the contents of the /proc/interrupts file displays something like the following:

```
# cat interrupts

           CPU0
  0:      32657          XT-PIC  timer
  1:       1063          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  8:          3          XT-PIC  rtc
  9:          0          XT-PIC  cmpci
 11:        332          XT-PIC  nvidia
 14:       5289          XT-PIC  ide0
 15:         13          XT-PIC  ide1
NMI:          0
ERR:          0
```

Each of the numbers down the left-hand column represents the interrupt that is in use. Examining the contents of the file dynamically gathers the associated data and displays it to the screen. Most of the /proc file system is read-only; however, some files allow kernel variable to be changed. This provides a mechanism to actually tune the kernel without recompiling and rebooting.

The procinfo utility summarizes /proc file system information into a display similar to the following:

```
# /usr/bin/procinfo

Linux 2.4.18 (root@DEB) (gcc 2.95.4 20011002 ) #2 1CPU [DEB.(none)]

Memory:      Total        Used        Free      Shared     Buffers      Cached
Mem:        513908      107404      406504           0        2832       82180
Swap:       265032           0      265032

Bootup: Sun Jan 19 14:58:27 2003    Load average: 0.29 0.13 0.05 1/30 566

user  :       0:00:10.26   2.3%  page in :    74545  disk 1:     6459r     796w
nice  :       0:00:00.00   0.0%  page out:     9416  disk 2:       19r       0w
system:       0:00:19.55   4.5%  swap in :        1
idle  :       0:06:48.30  93.2%  swap out:        0
uptime:       0:07:18.11         context :    22059

irq  0:     43811 timer                 irq  9:         0 cmpci
irq  1:      1427 keyboard              irq 11:       332 nvidia
irq  2:         0 cascade [4]           irq 12:         2
```

```
irq  6:         2                    irq 14:      7251 ide0
irq  8:         3 rtc                irq 15:        83 ide1
```

/proc/apm

Advanced power management info.

/proc/bus

Directory containing bus specific information.

/proc/cmdline

Kernel command line.

/proc/cpuinfo

Information about the processor, such as its type, make, model, and performance.

/proc/devices

List of device drivers configured into the currently running kernel (block and character).

/proc/dma

Shows which DMA channels are being used at the moment.

/proc/driver

Various drivers grouped here, currently rtc

/proc/execdomains

Execdomains, related to security.

/proc/fb

Frame Buffer devices.

/proc/filesystems

Filesystems configured/supported into/by the kernel.

/proc/fs

File system parameters, currently nfs/exports.

/proc/ide

This subdirectory contains information about all IDE devices of which the kernel is aware. There is one subdirectory for each IDE controller, the file drivers and a link for each IDE device, pointing to the device directory in the controller-specific subtree. The file drivers contains general information about the drivers used for the IDE devices. More detailed information can be found in the controller-specific subdirectories. These are named ide0, ide1 and so on. Each of these directories contains the files shown here:

/proc/ide/ide?/channel

IDE channel (0 or 1)

/proc/ide/ide?/config

Configuration (only for PCI/IDE bridge)

/proc/ide/ide?/mate

Mate name (onchip partnered controller)

/proc/ide/ide?/model

Type/Chipset of IDE controller

Each device connected to a controller has a separate subdirectory in the controllers directory. The following files listed are contained in these directories:

/proc/ide/ide?/model/cache

The cache.

/proc/ide/ide?/model/capacity

Capacity of the medium (in 512Byte blocks)

/proc/ide/ide?/model/driver

driver and version

/proc/ide/ide?/model/geometry

physical and logical geometry

/proc/ide/ide?/model/identify

device identify block

/proc/ide/ide?/model/media

media type

/proc/ide/ide?/model/model

device identifier

/proc/ide/ide?/model/settings

device setup

/proc/ide/ide?/model/smart_thresholds

IDE disk management thresholds

/proc/ide/ide?/model/smart_values

IDE disk management values

/proc/interrupts

Shows which interrupts are in use, and how many of each there have been.

You can, for example, check which interrupts are currently in use and what they are used for by looking in the file /proc/interrupts:

```
# cat /proc/interrupts


  CPU0 0: 8728810
  XT-PIC timer 1: 895
  XT-PIC keyboard 2:
  0 XT-PIC cascade 3: 531695
  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: 182918
  XT-PIC PS/2 Mouse 13: 1
  XT-PIC fpu 14: 1232265
  XT-PIC ide0 15: 7
  XT-PIC ide1 NMI: 0
```

In 2.4 based kernels a couple of lines were added to this file LOC & ERR (this is the output of an SMP machine):

```
# cat /proc/interrupts


  CPU0 CPU1
  0: 1243498 1214548 IO-APIC-edge timer
  1: 8949 8958 IO-APIC-edge keyboard
  2: 0 0 XT-PIC cascade
  5: 11286 10161 IO-APIC-edge soundblaster
  8: 1 0 IO-APIC-edge rtc
  9: 27422 27407 IO-APIC-edge 3c503
  12: 113645 113873 IO-APIC-edge PS/2 Mouse
  13: 0 0 XT-PIC fpu 14: 22491 24012 IO-APIC-edge ide0
  15: 2183 2415 IO-APIC-edge ide1
  17: 30564 30414 IO-APIC-level eth0
  18: 177 164 IO-APIC-level bttv NMI: 2457961 2457959
              LOC: 2457882 2457881 ERR: 2155
```

NMI is incremented in this case because every timer interrupt generates a NMI (Non Maskable Interrupt) which is used by the NMI Watchdog to detect lookups.

LOC is the local interrupt counter of the internal APIC of every CPU.

ERR is incremented in the case of errors in the IO-APIC bus (the bus that connects the CPUs in an SMP system. This means that an error has been detected, the IO-APIC automatically retries the transmission, so it should not be a big problem, but you should read the SMP-FAQ.

In this context it could be interesting to note the new irq directory in 2.4. It could be used to set IRQ to CPU affinity, this means that you can "hook" an IRQ to only one CPU, or to exclude a CPU from handling IRQs. The contents of the irq subdir is one subdir for each IRQ, and one file; prof_cpu_mask. For example,

```
# ls /proc/irq/ 0 10 12 14 16 18 2 4 6 8 prof_cpu_mask
                1 11 13 15 17 19 3 5 7 9



# ls /proc/irq/0/ smp_affinity
```

The contents of the prof_cpu_mask file and each smp_affinity file for each IRQ is the same by default:

```
# cat /proc/irq/0/smp_affinity
ffffffff
```

It's a bitmask, in which you can specify which CPUs can handle the IRQ, you can set it by doing:

```
# echo 1 > /proc/irq/prof_cpu_mask
```

This means that only the first CPU will handle the IRQ, but you can also echo 5 which means that only the first and fourth CPU can handle the IRQ. The way IRQs are routed is handled by the IO-APIC, and its Round Robin between all the CPUs which are allowed to handle it. As usual the kernel has more info than you and does a better job than you, so the defaults are the best choice for almost everyone.

/proc/iomem

Memory map.

/proc/ioports

Which I/O ports are in use at the moment.

/proc/irq

Masks for irq to cpu affinity.

/proc/isapnp

ISA PnP (Plug&Play) Info.

/proc/kcore

An image of the physical memory of the system (can be ELF or A.OUT (deprecated in 2.4)). This is exactly the same size as your physical memory, but does not really take up that much memory; it is generated on the fly as programs access it. (Remember: unless you copy it elsewhere, nothing under /proc takes up any disk space at all.)

/proc/kmsg

Messages output by the kernel. These are also routed to syslog.

/proc/ksyms

Kernel symbol table.

/proc/loadavg

The 'load average' of the system; three indicators of how much work the system has done during the last 1, 5 & 15 minutes.

/proc/locks

Kernel locks.

/proc/meminfo

Information about memory usage, both physical and swap. Concatenating this file produces similar results to using 'free' or the first few lines of 'top'.

/proc/misc

Miscellaneous pieces of information. This is for information that has no real place within the rest of the proc filesystem.

/proc/modules

Kernel modules currently loaded. Typically its output is the same as that given by the 'lsmod' command.

/proc/mounts

Mounted filesystems

/proc/mtrr

Information regarding mtrrs. (On Intel P6 family processors (Pentium Pro, Pentium II and later) the Memory Type Range Registers (MTRRs) may be used to control processor access to memory ranges. This is most useful when you have a video (VGA) card on a PCI or AGP bus. Enabling write-combining allows bus write transfers to be combined into a larger transfer before bursting over the PCI/AGP bus. This can increase performance of image write operations 2.5 times or more. The Cyrix 6x86, 6x86MX and M II processors have Address Range Registers (ARRs) which provide a similar functionality to MTRRs. For these, the ARRs are used to emulate the MTRRs. The AMD K6-2 (stepping 8 and above) and K6-3 processors have two MTRRs. These are supported. The AMD Athlon family provide 8 Intel style MTRRs. The Centaur C6 (WinChip) has 8 MCRs, allowing write-combining. These are also supported. The VIA Cyrix III and VIA C3 CPUs offer 8 Intel style MTRRs.) For more details regarding mtrr technology see /usr/src/linux/Documentation/mtrr.txt.

/proc/net

Status information about network protocols.

IPv6 information

/proc/net/udp6

UDP sockets (IPv6).

/proc/net/tcp6

TCP sockets (IPv6).

/proc/net/raw6

Raw device statistics (IPv6).

/proc/net/igmp6

IP multicast addresses, which this host joined (IPv6).

/proc/net/if_inet6

List of IPv6 interface addresses.

/proc/net/ipv6_route

Kernel routing table for IPv6.

/proc/net/rt6_stats

Global IPv6 routing tables statistics.

/proc/net/sockstat6

Socket statistics (IPv6).

/proc/net/snmp6

Snmp data (IPv6).

General Network information

/proc/net/arp

Kernel ARP table.

/proc/net/dev

network devices with statistics.

/proc/net/dev_mcast

the Layer2 multicast groups which a device is listening to (interface index, label, number of references, number of bound addresses).

/proc/net/dev_stat

network device status.

/proc/net/ip_fwchains

Firewall chain linkage.

/proc/net/ip_fwnames

Firewall chain names.

/proc/net/ip_masq

Directory containing the masquerading tables.

/proc/net/ip_masquerade

Major masquerading table.

/proc/net/netstat

Network statistics.

/proc/net/raw

raw device statistics.

/proc/net/route

Kernel routing table.

/proc/net/rpc

Directory containing rpc info.

/proc/net/rt_cache

Routing cache.

/proc/net/snmp

SNMP data.

/proc/net/sockstat

Socket statistics.

/proc/net/tcp

TCP sockets.

/proc/net/tr_rif

Token ring RIF routing table.

/proc/net/udp

UDP sockets.

/proc/net/unix

UNIX domain sockets.

/proc/net/wireless

Wireless interface data (Wavelan etc).

/proc/net/igmp

IP multicast addresses, which this host joined.

/proc/net/psched

Global packet scheduler parameters.

/proc/net/netlink

List of PF_NETLINK sockets.

/proc/net/ip_mr_vifs

List of multicast virtual interfaces.

/proc/net/ip_mr_cache

List of multicast routing cache.

You can use this information to see which network devices are available in your system and how much traffic was routed over those devices. In addition, each Channel Bond interface has its own directory. For example, the bond0 device will have a directory called /proc/net/bond0/. It will contain information that is specific to that bond, such as the current slaves of the bond, the link status of the slaves, and how many times the slaves link has failed.

/proc/parport

The directory /proc/parport contains information about the parallel ports of your system. It has one subdirectory for each port, named after the port number (0,1,2,...).

/proc/parport/autoprobe

Any IEEE-1284 device ID information that has been acquired.

/proc/parport/devices

list of the device drivers using that port. A + will appear by the name of the device currently using the port (it might not appear against any).

/proc/parport/hardware

Parallel port's base address, IRQ line and DMA channel.

/proc/parport/irq

IRQ that parport is using for that port. This is in a separate file to allow you to alter it by writing a new value in (IRQ number or none).

/proc/partitions

Table of partitions known to the system

/proc/pci, /proc/bus/pci

Depreciated info of PCI bus.

/proc/rtc

Real time clock

/proc/scsi

If you have a SCSI host adapter in your system, you'll find a subdirectory named after the driver for this adapter in /proc/scsi. You'll also see a list of all recognized SCSI devices in /proc/scsi. The directory named after the driver has one file for each adapter found in the system. These files contain information about the controller, including the used IRQ and the IO address range. The amount of information shown is dependent on the adapter you use.

/proc/self

A symbolic link to the process directory of the program that is looking at /proc. When two processes look at /proc, they get different links. This is mainly a convenience to make it easier for programs to get at their process directory.

/proc/slabinfo

The slabinfo file gives information about memory usage at the slab level. Linux uses slab pools for memory management above page level in version 2.2. Commonly used objects have their own slab pool (such as network buffers, directory cache, and so on).

/proc/stat

Overall/various statistics about the system, such as the number of page faults since the system was booted.

/proc/swaps

Swap space utilization

/proc/sys

This is not only a source of information, it also allows you to change parameters within the kernel without the need for recompilation or even a system reboot. Take care when attempting this as it can both optimize your system and also crash it. It is advisable to read both documentation and source before actually making adjustments. The entries in /proc may change slightly between kernel versions, so if there is any doubt review the kernel documentation in the directory /usr/src/linux/Documentation. Under some circumstances, you may have no alternative but to reboot the machine once an error occurs. To change a value, simply echo the new value into the file. An example is given below in the section on the file system data. Of course, you need to be 'root' to do any of this. You can create your own boot script to perform this every time your system boots.

/proc/sys/fs

Contains file system data. This subdirectory contains specific file system, file handle, inode, dentry and quota information.

dentry-state

Status of the directory cache. Since directory entries are dynamically allocated and deallocated, this file indicates the current status. It holds six values, in which the last two are not used and are always zero. The others are listed below:

```
File      Content
nr_dentry Almost always zero
nr_unused Number of unused cache entries
age_limit in seconds after the entry may be
          reclaimed, when memory is short want_pages internally
```

dquot-max

The file dquot-max shows the maximum number of cached disk quota entries.

dquot-nr

shows the number of allocated disk quota entries and the number of free disk quota entries. If the number of available cached disk quotas is very low and you have a large number of simultaneous system users, you might want to raise the limit.

file-nr and file-max

The kernel allocates file handles dynamically, but doesn't free them again at this time. The value in file-max denotes the maximum number of file handles that the Linux kernel will allocate. When you get a lot of error messages about running out of file handles, you might want to raise this limit. The default value is 4096. To change it, just write the new number into the file:

```
# cat /proc/sys/fs/file-max
4096
# echo 8192 > /proc/sys/fs/file-max
# cat /proc/sys/fs/file-max
8192
```

This method of revision is useful for all customizable parameters of the kernel - simply echo the new value to the corresponding file.

The three values in file-nr denote the number of allocated file handles, the number of used file handles, and the maximum number of file handles. When the allocated file handles come close to the maximum, but the number of actually used handles is far behind, you've encountered a peak in your usage of file handles and you don't need to increase the maximum.

inode-state, inode-nr and inode-max

As with file handles, the kernel allocates the inode structures dynamically, but can't free them yet.

The value in inode-max denotes the maximum number of inode handlers. This value should be 3 to 4 times larger than the value in file-max, since stdin, stdout, and network sockets also need an inode struct to handle them. If you regularly run out of inodes, you should increase this value.

The file inode-nr contains the first two items from inode-state, so we'll skip to that file...

inode-state contains three actual numbers and four dummy values. The numbers are nr_inodes, nr_free_inodes, and preshrink (in order of appearance).

nr_inodes

Denotes the number of inodes the system has allocated. This can be slightly more than inode-max because Linux allocates them one pageful at a time.

nr_free_inodes

Represents the number of free inodes and preshrink is nonzero when nr_inodes is greater than inode-max and the system needs to prune the inode list instead of allocating more.

super-nr and super-max

Again, super block structures are allocated by the kernel, but not freed. The file super-max contains the maximum number of super block handlers, where super-nr shows the number of currently allocated ones. Every mounted file system needs a super block, so if you plan to mount lots of file systems, you may want to increase these numbers.

binfmt_misc

This handles the kernel support for miscellaneous binary formats. binfmt_misc provides the ability to register additional binary formats to the kernel without compiling an additional module/kernel. Therefore, binfmt_misc needs to know magic numbers at the beginning or the filename extension of the binary. It works by maintaining a linked list of structs that contain a description of a binary format, including a magic with size (or the filename extension), offset and mask, and the interpreter name. On request it invokes the given interpreter with the original program as argument, as binfmt_java and binfmt_em86 and binfmt_mz do. Since binfmt_misc does not define any default binary-formats, you have to register an additional binary-format. There are two general files in binfmt_misc and one file per registered format. The two general files are register and status. To register a new binary format you have to issue the command echo :name:type:offset:magic:mask:interpreter: > /proc/sys/fs/binfmt_misc/register with appropriate name (the name for the /proc-dir entry), offset (defaults to 0, if omitted), magic, mask (which can be omitted, defaults to all 0xff) and last but not least, the interpreter that is to be invoked (for example and testing /bin/echo). Type can be M for usual magic matching or E for filename extension matching (give extension in place of magic). If you do a cat on the file /proc/sys/fs/binfmt_misc/status, you will get the current status (enabled/disabled) of binfmt_misc. Change the status by echoing 0 (disables) or 1 (enables) or -1 (caution: this clears all previously registered binary formats) to status. For example echo 0 > status to disable binfmt_misc (temporarily). Each registered handler has an entry in /proc/sys/fs/binfmt_misc. These files perform the same function as status, but their scope is limited to the actual binary format. By 'cating' this file, you also receive all related information about the interpreter/magic of the binfmt. An example of the usage of binfmt_misc (emulate binfmt_java) follows:

```
cd /proc/sys/fs/binfmt_misc
echo ':Java:M::\xca\xfe\xba\xbe::/usr/local/java/bin/javawrapper:'
> register
echo ':HTML:E::html::/usr/local/java/bin/appletviewer:'
> register
echo ':Applet:M::<!--applet::/usr/local/java/bin/appletviewer:' >
register
echo ':DEXE:M::\x0eDEX::/usr/bin/dosexec:' < register
```

These four lines add support for Java executables and Java applets (like binfmt_java, additionally recognizing the .html extension with no need to put <!--applet> to every applet file). You have to install the JDK and the shell-script /usr/local/java/bin/javawrapper too. It works around the brokenness of the Java filename handling. To add a Java binary, just create a link to the class-file somewhere in the path.

/proc/sys/kernel

This directory reflects general kernel behaviors and the contents will be dependent upon your configuration. Here you'll find the most important files, along with descriptions of what they mean and how to use them.

/proc/sys/kernel/acct

The file contains three values; highwater, lowwater, and frequency. It exists only when BSD-style process accounting is enabled. These values control its behavior. If the free space on the file system

where the log lives goes below lowwater percentage, accounting suspends. If it goes above highwater percentage, accounting resumes. Frequency determines how often you check the amount of free space (value is in seconds). Default settings are: 4, 2, and 30. That is, suspend accounting if there is less than 2 percent free; resume it if we have a value of 3 or more percent; consider information about the amount of free space valid for 30 seconds

/proc/sys/kernel/ctrl-alt-del

When the value in this file is 0, ctrl-alt-del is trapped and sent to the init program to handle a graceful restart. However, when the value is greater that zero, Linux's reaction to this key combination will be an immediate reboot, without syncing its dirty buffers. It should be noted that when a program (like dosemu) has the keyboard in raw mode, the ctrl-alt-del is intercepted by the program before it ever reaches the kernel tty layer, and it is up to the program to decide what to do with it.

/proc/sys/kernel/domainname, /proc/sys/kernel/hostname

These files can be controlled to set the NIS domainname and hostname of your box. For the classic darkstar.frop.org a simple: # echo "darkstar" > /proc/sys/kernel/hostname # echo "frop.org" > /proc/sys/kernel/domainname would suffice to set your hostname and NIS domainname. /proc/sys/kernel/osrelease, /proc/sys/kernel/ostype, /proc/sys/kernel/version The names make it pretty obvious what these fields contain: # cat /proc/sys/kernel/osrelease 2.2.12 # cat /proc/sys/kernel/ostype Linux # cat /proc/sys/kernel/version #4 Fri Oct 1 12:41:14 PDT 1999 The files osrelease and ostype should be clear enough. Version needs a little more clarification. The #4 means that this is the 4th kernel built from this source base and the date after it indicates the time the kernel was built. The only way to tune these values is to rebuild the kernel.

/proc/sys/kernel/panic

The value in this file represents the number of seconds the kernel waits before rebooting on a panic. When you use the software watchdog, the recommended setting is 60. If set to 0, the auto reboot after a kernel panic is disabled, which is the default setting.

/proc/sys/kernel/printk

The four values in printk denote * console_loglevel, * default_message_loglevel, * minimum_console_level and * default_console_loglevel respectively. These values influence printk() behavior when printing or logging error messages, which come from inside the kernel. See syslog(2) for more information on the different log levels.

/proc/sys/kernel/console_loglevel

Messages with a higher priority than this will be printed to the console.

/proc/sys/kernel/default_message_level

Messages without an explicit priority will be printed with this priority.

/proc/sys/kernel/minimum_console_loglevel

Minimum (highest) value to which the console_loglevel can be set.

/proc/sys/kernel/default_console_loglevel

Default value for console_loglevel.

/proc/sys/kernel/sg-big-buff

This file shows the size of the generic SCSI (sg) buffer. At this point, you can't tune it yet, but you can change it at compile time by editing include/scsi/sg.h and changing the value of SG_BIG_BUFF. If you use a scanner with SANE (Scanner Access Now Easy) you might want to set this to a higher value. Refer to the SANE documentation on this issue.

/proc/sys/kernel/modprobe

The location where the modprobe binary is located. The kernel uses this program to load modules on demand.

/proc/sys/vm

The files in this directory can be used to tune the operation of the virtual memory (VM) subsystem of the Linux kernel. In addition, one of the files (bdflush) has some influence on disk usage.

nfract

This parameter governs the maximum number of dirty buffers in the buffer cache. Dirty means that the contents of the buffer still have to be written to disk (as opposed to a clean buffer, which can just be forgotten about). Setting this to a higher value means that Linux can delay disk writes for a long time, but it also means that it will have to do a lot of I/O at once when memory becomes short. A lower value will spread out disk I/O more evenly.

ndirty

Ndirty gives the maximum number of dirty buffers that bdflush can write to the disk at one time. A high value will mean delayed, bursty I/O, while a small value can lead to memory shortage when bdflush isn't woken up often enough.

nrefill

This is the number of buffers that bdflush will add to the list of free buffers when refill_freelist() is called. It is necessary to allocate free buffers beforehand, since the buffers are often different sizes than the memory pages and some bookkeeping needs to be done beforehand. The higher the number, the more memory will be wasted and the less often refill_freelist() will need to run.

nref_dirt

When refill_freelist() comes across more than nref_dirt dirty buffers, it will wake up bdflush.

age_buffer, age_super

Finally, the age_buffer and age_super parameters govern the maximum time Linux waits before writing out a dirty buffer to disk. The value is expressed in jiffies (clockticks), the number of jiffies per second is 100. Age_buffer is the maximum age for data blocks, while age_super is for filesystems meta data.

buffermem

The three values in this file control how much memory should be used for buffer memory. The percentage is calculated as a percentage of total system memory.

The values are:

min_percent

This is the minimum percentage of memory that should be spent on buffer memory.

borrow_percent

When Linux is short on memory, and the buffer cache uses more than it has been allotted, the memory management (MM) subsystem will prune the buffer cache more heavily than other memory to compensate.

max_percent

This is the maximum amount of memory that can be used for buffer memory.

freepages

This file contains three values: min, low and high:

min

When the number of free pages in the system reaches this number, only the kernel can allocate more memory.

low

If the number of free pages falls below this point, the kernel starts swapping aggressively.

high

The kernel tries to keep up to this amount of memory free; if memory falls below this point, the kernel starts gently swapping in the hopes that it never has to do really aggressive swapping.

kswapd

Kswapd is the kernel swap out daemon. That is, kswapd is that piece of the kernel that frees memory when it gets fragmented or full. Since every system is different, you'll probably want some control over this piece of the system.

The file contains three numbers:

tries_base

The maximum number of pages kswapd tries to free in one round is calculated from this number. Usually this number will be divided by 4 or 8 (see mm/vmscan.c), so it isn't as big as it looks. When you need to increase the bandwidth to/from swap, you'll want to increase this number.

tries_min

This is the minimum number of times kswapd tries to free a page each time it is called. Basically it's just there to make sure that kswapd frees some pages even when it's being called with minimum priority.

swap_cluster

This is probably the greatest influence on system performance. swap_cluster is the number of pages kswapd writes in one turn. You'll want this value to be large so that kswapd does its I/O in large chunks and the disk doesn't have to seek as often, but you don't want it to be too large since that would flood the request queue.

overcommit_memory

This file contains one value. The following algorithm is used to decide if there's enough memory: if the value of overcommit_memory is positive, then there's always enough memory. This is a useful feature, since programs often malloc() huge amounts of memory 'just in case', while they only use a small part of it. Leaving this value at 0 will lead to the failure of such a huge malloc(), when in fact the system has enough memory for the program to run. On the other hand, enabling this feature can cause you to run out of memory and thrash the system to death, so large and/or important servers will want to set this value to 0.

pagecache

This file does exactly the same job as buffermem, only this file controls the amount of memory allowed for memory mapping and generic caching of files. You don't want the minimum level to be too low, otherwise your system might thrash when memory is tight or fragmentation is high.

pagetable_cache

The kernel keeps a number of page tables in a per-processor cache (this helps a lot on SMP systems). The cache size for each processor will be between the low and the high value. On a low-memory, single CPU system, you can safely set these values to 0 so you don't waste memory. It is used on SMP systems so that the system can perform fast pagetable allocations without having to acquire the kernel memory lock. For large systems, the settings are probably fine. For normal systems they won't hurt a bit. For small systems (less than 16MB ram) it might be advantageous to set both values to 0.

swapctl

This file contains no less than 8 variables. All of these values are used by kswapd. The first four variables sc_max_page_age, sc_page_advance, sc_page_decline and sc_page_initial_age are used to keep track of Linux's page aging. Page ageing is a bookkeeping method to track which pages of memory are often used, and which pages can be swapped out without consequences.

When a page is swapped in, it starts at sc_page_initial_age (default 3) and when the page is scanned by kswapd, its age is adjusted according to the following scheme.

If the page was used since the last time we scanned, its age is increased by sc_page_advance (default 3). Where the maximum value is given by sc_max_page_age (default 20). Otherwise (meaning it wasn't used) its age is decreased by sc_page_decline (default 1).

When a page reaches age 0, it's ready to be swapped out.

The variables sc_age_cluster_fract, sc_age_cluster_min, sc_pageout_weight and sc_bufferout_weight, can be used to control kswapd's aggressiveness in swapping out pages.

Sc_age_cluster_fract is used to calculate how many pages from a process are to be scanned by kswapd. The formula used is

(sc_age_cluster_fract divided by 1024) times resident set size

So if you want kswapd to scan the whole process, sc_age_cluster_fract needs to have a value of 1024. The minimum number of pages kswapd will scan is represented by sc_age_cluster_min, which is done so that kswapd will also scan small processes. The values of sc_pageout_weight and sc_bufferout_weight are used to control how many tries kswapd will make in order to swap out one page/buffer. These values can be used to fine-tune the ratio between user pages and buffer/cache memory. When you find that your Linux system is swapping out too many process pages in order to

satisfy buffer memory demands, you may want to either increase sc_bufferout_weight, or decrease the value of sc_pageout_weight.

/proc/sys/dev

Device specific parameters. Currently there is only support for CDROM drives, and for those, there is only one read-only file containing information about the CD-ROM drives attached to the system: >cat /proc/sys/dev/cdrom/info CD-ROM information, Id: cdrom.c 2.55 1999/04/25 drive name: sr0 hdb drive speed: 32 40 drive # of slots: 1 0 Can close tray: 1 1 Can open tray: 1 1 Can lock tray: 1 1 Can change speed: 1 1 Can select disk: 0 1 Can read multisession: 1 1 Can read MCN: 1 1 Reports media changed: 1 1 Can play audio: 1 1 You see two drives, sr0 and hdb, along with a list of their features.

SUNRPC

/proc/sys/sunrpc

This directory contains four files, which enable or disable debugging for the RPC functions NFS, NFS-daemon, RPC and NLM. The default values are 0. They can be set to one to turn debugging on. (The default value is 0 for each)

/proc/sys/net

The interface to the networking parts of the kernel is located in /proc/sys/net. The following table shows all possible subdirectories. You may see only some of them, depending on your kernel's configuration. Our main focus will be on IP networking since AX15, X.25, and DEC Net are only minor players in the Linux world. Should you wish review the online documentation and the kernel source to get a detailed view of the parameters for those protocols not covered here. In this section we'll discuss the subdirectories listed above. As default values are suitable for most needs, there is no need to change these values.

GENERAL PARAMETERS

/proc/sys/net/core

Network core options

rmem_default

The default setting of the socket receive buffer in bytes.

rmem_max

The maximum receive socket buffer size in bytes.

wmem_default

The default setting (in bytes) of the socket send buffer.

wmem_max

The maximum send socket buffer size in bytes.

message_burst and message_cost

These parameters are used to limit the warning messages written to the kernel log from the networking code. They enforce a rate limit to make a denial-of-service attack impossible. A higher

message_cost factor, results in fewer messages that will be written. Message_burst controls when messages will be dropped. The default settings limit warning messages to one every five seconds.

netdev_max_backlog

Maximum number of packets, queued on the INPUT side, when the interface receives packets faster than kernel can process them.

optmem_max

Maximum ancillary buffer size allowed per socket. Ancillary data is a sequence of struct cmsghdr structures with appended data.

UNIX DOMAIN SOCKETS

/proc/sys/net/unix

Parameters for Unix domain sockets

There are only two files in this subdirectory. They control the delays for deleting and destroying socket descriptors.

IPv4

/proc/sys/net/ipv4

IPV4 settings. IP version 4 is still the most used protocol in Unix networking. It will be replaced by IP version 6 in the next couple of years, but for the moment it's the de facto standard for the internet and is used in most networking environments around the world. Because of the importance of this protocol, we'll have a deeper look into the subtree controlling the behavior of the Ipv4 subsystem of the Linux kernel.

Let's start with the entries in /proc/sys/net/ipv4.

ICMP settings

icmp_echo_ignore_all and icmp_echo_ignore_broadcasts

Turn on (1) or off (0), if the kernel should ignore all ICMP ECHO requests, or just those to broadcast and multicast addresses.

Please note that if you accept ICMP echo requests with a broadcast/multi\-cast destination address your network may be used as an exploder for denial of service packet flooding attacks to other hosts.

icmp_destunreach_rate, icmp_echoreply_rate, icmp_paramprob_rate and icmp_timeexeed_rate

Sets limits for sending ICMP packets to specific targets. A value of zero disables all limiting. Any positive value sets the maximum package rate in hundredth of a second (on Intel systems).

IP settings

ip_autoconfig

This file contains the number one if the host received its IP configuration by RARP, BOOTP, DHCP or a similar mechanism. Otherwise it is zero.

ip_default_ttl

TTL (Time To Live) for IPv4 interfaces. This is simply the maximum number of hops a packet may travel.

ip_dynaddr

Enable dynamic socket address rewriting on interface address change. This is useful for dialup interface with changing IP addresses.

ip_forward

Enable or disable forwarding of IP packages between interfaces. Changing this value resets all other parameters to their default values. They differ if the kernel is configured as host or router.

ip_local_port_range

Range of ports used by TCP and UDP to choose the local port. Contains two numbers, the first number is the lowest port, the second number the highest local port. Default is 1024-4999. Should be changed to 32768-61000 for high-usage systems.

ip_no_pmtu_disc

Global switch to turn path MTU discovery off. It can also be set on a per socket basis by the applications or on a per route basis.

ip_masq_debug

Enable/disable debugging of IP masquerading.

IP fragmentation settings

ipfrag_high_trash and ipfrag_low_trash

Maximum memory used to reassemble IP fragments. When ipfrag_high_thrash bytes of memory is allocated for this purpose, the fragment handler will toss packets until ipfrag_low_thrash is reached.

ipfrag_time

Time in seconds to keep an IP fragment in memory.

TCP settings

tcp_ecn

This file controls the use of the ECN bit in the IPv4 headers, this is a new feature about Explicit Congestion Notification, but some routers and firewalls block traffic that has this bit set, so it could be necessary to echo 0 to /proc/sys/net/ipv4/tcp_ecn, if you want to talk to this sites. For more info you could read RFC2481.

tcp_retrans_collapse

Bug-to-bug compatibility with some broken printers. On retransmit, try to send larger packets to work around bugs in certain TCP stacks. Can be turned off by setting it to zero.

tcp_keepalive_probes

Number of keep alive probes TCP sends out, until it decides that the connection is broken.

tcp_keepalive_time

How often TCP sends out keep alive messages, when keep alive is enabled. The default is 2 hours.

tcp_syn_retries

Number of times initial SYNs for a TCP connection attempt will be retransmitted. Should not be higher than 255. This is only the timeout for outgoing connections, for incoming connections the number of retransmits is defined by tcp_retries1.

tcp_sack

Enable select acknowledgments after RFC2018.

tcp_timestamps

Enable timestamps as defined in RFC1323.

tcp_stdurg

Enable the strict RFC793 interpretation of the TCP urgent pointer field. The default is to use the BSD compatible interpretation of the urgent pointer pointing to the first byte after the urgent data. The RFC793 interpretation is to have it point to the last byte of urgent data. Enabling this option may lead to interoperability problems. Disabled by default.

tcp_syncookies

Only valid when the kernel was compiled with CONFIG_SYNCOOKIES. Send out syncookies when the syn backlog queue of a socket overflows. This is to ward off the common 'syn flood attack'. Disabled by default. Note that the concept of a socket backlog is abandoned. This means the peer may not receive reliable error messages from an over loaded server with syncookies enabled.

tcp_window_scaling

Enable window scaling as defined in RFC1323.

tcp_fin_timeout

The length of time in seconds it takes to receive a final FIN before the socket is always closed. This is strictly a violation of the TCP specification, but required to prevent denial-of-service attacks.

tcp_max_ka_probes

Indicates how many keep alive probes are sent per slow timer run. Should not be set too high to prevent bursts.

tcp_max_syn_backlog

Length of the per socket backlog queue. Since Linux 2.2 the backlog specified in listen(2) only specifies the length of the backlog queue of already established sockets. When more connection requests arrive Linux starts to drop packets. When syncookies are enabled the packets are still answered and the maximum queue is effectively ignored.

tcp_retries1

Defines how often an answer to a TCP connection request is retransmitted before giving up.

tcp_retries2

Defines how often a TCP packet is retransmitted before giving up.

/proc/sys/net/ipv4/conf

Here you'll find one subdirectory for each interface the system knows about and one directory called all. Changes in the all subdirectory affect all interfaces, whereas changes in the other subdirectories affect only one interface. All directories have the same entries:

accept_redirects

This switch decides if the kernel accepts ICMP redirect messages or not. The default is 'yes' if the kernel is configured for a regular host and 'no' for a router configuration.

accept_source_route

Should source routed packages be accepted or declined. The default is dependent on the kernel configuration. It's 'yes' for routers and 'no' for hosts.

bootp_relay

Accept packets with source address 0.b.c.d with destinations not to this host as local ones. It is supposed that a BOOTP relay daemon will catch and forward such packets. The default is 0.

forwarding

Enable or disable IP forwarding on this interface.

log_martians

Log packets with source addresses with no known route to kernel log.

mc_forwarding

Do multicast routing. The kernel needs to be compiled with CONFIG_MROUTE and a multicast routing daemon is required.

proxy_arp

Does (1) or does not (0) perform proxy ARP.

rp_filter

Integer value determines if a source validation should be made. 1 means yes, 0 means no. Disabled by default, but local/broadcast address spoofing is always on. If you set this to 1 on a router that is the only connection for a network to the net, it will prevent spoofing attacks against your internal networks (external addresses can still be spoofed), without the need for additional firewall rules.

secure_redirects

Accept ICMP redirect messages only for gateways, listed in default gateway list. Enabled by default.

shared_media

If it is not set the kernel does not assume that different subnets on this device can communicate directly. Default setting is 'yes'.

send_redirects

Determines whether to send ICMP redirects to other hosts.

Routing settings

The directory /proc/sys/net/ipv4/route contains several file to control routing issues.

error_burst and error_cost

These parameters are used to limit the warning messages written to the kernel log from the routing code. The higher the error_cost factor is, the fewer messages will be written. Error_burst controls when messages will be dropped. The default settings limit warning messages to one every five seconds.

flush

Writing to this file results in a flush of the routing cache.

gc_elastic, gc_interval, gc_min_interval, gc_tresh, gc_timeout

Values to control the frequency and behavior of the garbage collection algorithm for the routing cache.

max_size

Maximum size of the routing cache. Old entries will be purged once the cache reached has this size.

max_delay, min_delay

Delays for flushing the routing cache.

redirect_load, redirect_number

Factors which determine if more ICPM redirects should be sent to a specific host. No redirects will be sent once the load limit or the maximum number of redirects has been reached.

redirect_silence

Timeout for redirects. After this period redirects will be sent again, even if this has been stopped, because the load or number limit has been reached.

/proc/sys/net/ipv4/neigh

Network Neighbor handling. It contains settings about how to handle connections with direct neighbors (nodes attached to the same link). As we saw it in the conf directory, there is a default subdirectory which holds the default values, and one directory for each interface. The contents of the directories are identical, with the single exception that the default settings contain additional options to set garbage collection parameters.

In the interface directories you'll find the following entries:

base_reachable_time

A base value used for computing the random reachable time value as specified in RFC2461.

retrans_time

The time, expressed in jiffies (1/100 sec), between retransmitted Neighbor Solicitation messages. Used for address resolution and to determine if a neighbor is unreachable.

unres_qlen

Maximum queue length for a pending arp request - the number of packets which are accepted from other layers while the ARP address is still resolved.

anycast_delay

Maximum for random delay of answers to neighbor solicitation messages in jiffies (1/100 sec). Not yet implemented (Linux does not have anycast support yet).

ucast_solicit

Maximum number of retries for unicast solicitation.

mcast_solicit

Maximum number of retries for multicast solicitation.

delay_first_probe_time

Delay for the first time probe if the neighbor is reachable. (see gc_stale_time)

locktime

An ARP/neighbor entry is only replaced with a new one if the old is at least locktime old. This prevents ARP cache thrashing.

proxy_delay

Maximum time (real time is random [0..proxytime]) before answering to an ARP request for which we have an proxy ARP entry. In some cases, this is used to prevent network flooding.

proxy_qlen

Maximum queue length of the delayed proxy arp timer. (see proxy_delay).

app_solcit

Determines the number of requests to send to the user level ARP daemon. Use 0 to turn off.

gc_stale_time

Determines how often to check for stale ARP entries. After an ARP entry is stale it will be resolved again (which is useful when an IP address migrates to another machine). When ucast_solicit is greater than 0 it first tries to send an ARP packet directly to the known host When that fails and mcast_solicit is greater than 0, an ARP request is broadcasted.

APPLETALK

/proc/sys/net/appletalk

Holds the Appletalk configuration data when Appletalk is loaded. The configurable parameters are:

aarp-expiry-time

The amount of time we keep an ARP entry before expiring it. Used to age out old hosts.

aarp-resolve-time

The amount of time we will spend trying to resolve an Appletalk address.

aarp-retransmit-limit

The number of times we will retransmit a query before giving up.

aarp-tick-time

Controls the rate at which expires are checked.

/proc/net/appletalk

Holds the list of active Appletalk sockets on a machine. The fields indicate the DDP type, the local address (in network:node format) the remote address, the size of the transmit pending queue, the size of the received queue (bytes waiting for applications to read) the state and the uid owning the socket.

/proc/net/atalk_iface

lists all the interfaces configured for appletalk. It shows the name of the interface, its Appletalk address, the network range on that address (or network number for phase 1 networks), and the status of the interface.

/proc/net/atalk_route

lists each known network route. It lists the target (network) that the route leads to, the router (may be directly connected), the route flags, and the device the route is using.

IPX

The IPX protocol has no tunable values in proc/sys/net, it does, however, provide proc/net/ipx. This lists each IPX socket giving the local and remote addresses in Novell format (that is network:node:port). In accordance with the strange Novell tradition, everything but the port is in hex. Not_Connected is displayed for sockets that are not tied to a specific remote address. The Tx and Rx queue sizes indicate the number of bytes pending for transmission and reception. The state indicates the state the socket is in and the uid is the owning uid of the socket.

ipx_interface

Lists all IPX interfaces. For each interface it gives the network number, the node number, and indicates if the network is the primary network. It also indicates which device it is bound to (or Internal for internal networks) and the Frame Type if appropriate. Linux supports 802.3, 802.2, 802.2 SNAP and DIX (Blue Book) ethernet framing for IPX.

ipx_route

Table holding a list of IPX routes. For each route it gives the destination network, the router node (or Directly) and the network address of the router (or Connected) for internal networks.

/proc/sysvipc

Info of SysVIPC Resources (msg, sem, shm) (2.4)

/proc/tty

Information about the available and actually used tty's can be found in the directory /proc/tty. You'll find entries for drivers and line disciplines in this directory.

/proc/tty/drivers

list of drivers and their usage.

/proc/tty/ldiscs

registered line disciplines.

/proc/tty/driver/serial

usage statistic and status of single tty lines.

To see which tty's are currently in use, you can simply look into the file /proc/tty/drivers:

```
# cat /proc/tty/drivers
serial               /dev/cua        5  64-127 serial:callout
serial               /dev/ttyS       4  64-127 serial
pty_slave            /dev/pts      143   0-255 pty:slave
pty_master           /dev/ptm      135   0-255 pty:master
pty_slave            /dev/pts      142   0-255 pty:slave
pty_master           /dev/ptm      134   0-255 pty:master
pty_slave            /dev/pts      141   0-255 pty:slave
pty_master           /dev/ptm      133   0-255 pty:master
pty_slave            /dev/pts      140   0-255 pty:slave
pty_master           /dev/ptm      132   0-255 pty:master
pty_slave            /dev/pts      139   0-255 pty:slave
pty_master           /dev/ptm      131   0-255 pty:master
pty_slave            /dev/pts      138   0-255 pty:slave
pty_master           /dev/ptm      130   0-255 pty:master
pty_slave            /dev/pts      137   0-255 pty:slave
pty_master           /dev/ptm      129   0-255 pty:master
pty_slave            /dev/pts      136   0-255 pty:slave
pty_master           /dev/ptm      128   0-255 pty:master
pty_slave            /dev/ttyp       3   0-255 pty:slave
pty_master           /dev/pty        2   0-255 pty:master
/dev/vc/0            /dev/vc/0       4       0 system:vtmaster
/dev/ptmx            /dev/ptmx       5       2 system
/dev/console         /dev/console    5       1 system:console
/dev/tty             /dev/tty        5       0 system:/dev/tty
unknown              /dev/vc/%d      4    1-63 console
```

Note that while the above files tend to be easily readable text files, they can sometimes be formatted in a way that is not easily digestible. There are many commands that do little more than read the above files and format them for easier understanding. For example, the free program reads /proc/meminfo and converts the amounts given in bytes to kilobytes (and adds a little more information, as well).

/proc/uptime

The time the system has been up.

/proc/version

The kernel version.

/proc/video

BTTV info of video resources.

Prev /opt | Home | Up | Next /root