Previous Chapter - Table of Contents - Next Chapter

Chapter 11 System Monitoring - Notes

11.3 Learning Objectives:

- Understand the concept of inventory and gain familiarity with available system monitoring tools.
- Understand where the system stores log files and examine the most important ones.
- Use the `/proc` and `/sys` pseudo-filesystems.
- Use **sar** to father system activity and performance data and create reports that are readable by humans.

11.4 Available Monitoring Tools

Linux distributions comes with many standard performance/profiling tools already installed. Many familiar from other UNIX-like operating systems, while some developed specifically for Linux.

Most tools make use of mounted **pseudo-filesystems**, especially `/proc` and `/sys`, both which have already been discussed while examining filesystems/kernel configuration. Will look at both.

Also a number of graphical system monitors that hide many details, but will only consider command line tools in course.

Before considering main utilities in detail, can see summary on next few sections, broken down by type: note: some utilities have overlapping domains of coverage. Will revisit tables in following chapters that focus on specific topics.

Summary of main process monitoring utility tools:

Process and Load Monitoring Utilities

Utility	Purpose	Package
top	Process activity, dynamically updated	**procps**
uptime	How long system is running and average load	**procps**
ps	Detailed information about processes	**procps**
pstree	Tree of processes and their connections	**psmisc** (or **pstree**)
mpstat	Multiple processor usage	**sysstat**
iostat	CPU utilization and I/O statistics	**sysstat**
sar	Display and collect information about system activity	**sysstat**
numstat	Information about **NUMA** (Non-Uniform Memory Architecture)	**numactl**
strace	Information about all system calls a process makes	**strace**

Memory Monitoring Utilities

Utility	Purpose	Package
free	Brief summary of memory usage	**procps**

vmstat	Detailed virtual memory statistics and block I/O, dynamically updated	**procps**
pmap	Process memory map	**procps**

I/O Monitoring Utilities

Utility	Purpose	Package
iostat	CPU utilization and I/O statistics	**sysstat**
sar	Display and collect information about system activity	**sysstat**
vmstat	Detailed virtual memory statistics and block I/O, dynamically updated	**procps**

Network Monitoring Utilities

Utility	Purpose	Package
netstat	Detailed networking statistics	**netstat**
iptraf	Gather information on network interfaces	**iptraf**
tcpdump	Detailed analysis of network packets and traffic	**tcpdump**
wireshark	Detailed network traffic analysis	**wireshark**

11.5 System Log Files

System log files -> essential for monitoring/troubleshooting. In Linux, messages appear in various files under `/var/log`. Exact names vary with Linux distribution.

Ultimate control of how messages dealt with -> controlled by **syslogd** (usually **rsyslogd** on modern systems) daemon, common to many UNIX-like operating systems. Newer **systemd**-based systems can use **journalctl** instead,but usually retain **syslogd** and cooperate with it.

Important messages sent not only to logging files, but also to system console window. If not running **X**, or are at virtual terminal, will see them directly there as well. In addition, messages will be copied to `/var/log` messages (or to `/var/log/syslog` on Ubuntu), but if running **X**, have to take some steps to view them.

Can view new messages continuously as new lines appear with:

```
$ sudo tail -f /var/log/messages (or /var/log/syslog)
```

or

```
$ dmesg -w
```

which shows only kernel-related messages.

11.6 Important Log Files in /var/log

Besides looking at log messages in terminal window, can see them using graphical interfaces.

On GNOME desktop, can also access messages by clicking on `System -> Administration -> System Log` or `Applications -> System Tools -> Log File` View er in your Desktop menus, and other desktops have similar links you can locate.

Some important log files found under `/var/log`:

File	Purpose
`boot.log`	System boot messages
`dmesg`	Kernel messages saved after boot. To see current contents of kernel message buffer, type **dmesg**.
`messages` or `syslog`	All important system messages
`secure`	Security related messages

In order to keep log files from grow ing w ithout bound, **logrotate** program run periodically, keeps four previous copies (by default) of log files (optionally compressed). Controlled by `/etc/logrotate.conf`.

11.7 The /proc and /sys Pseudo-filesystems

`/proc` and `/sys` pseudo-filesystems contain lot of information about system. Many entries in these directory trees w ritable, can be used to change system behavior. Most cases, requires **root** user.

Pseudo-filesystems because totally exist in memory. If look at disk partition w hen system not running, there w ill be only empty directory w hich is used as mount point.

Information displayed is gathered only w hen looked at. No constant/periodic polling to update entries.

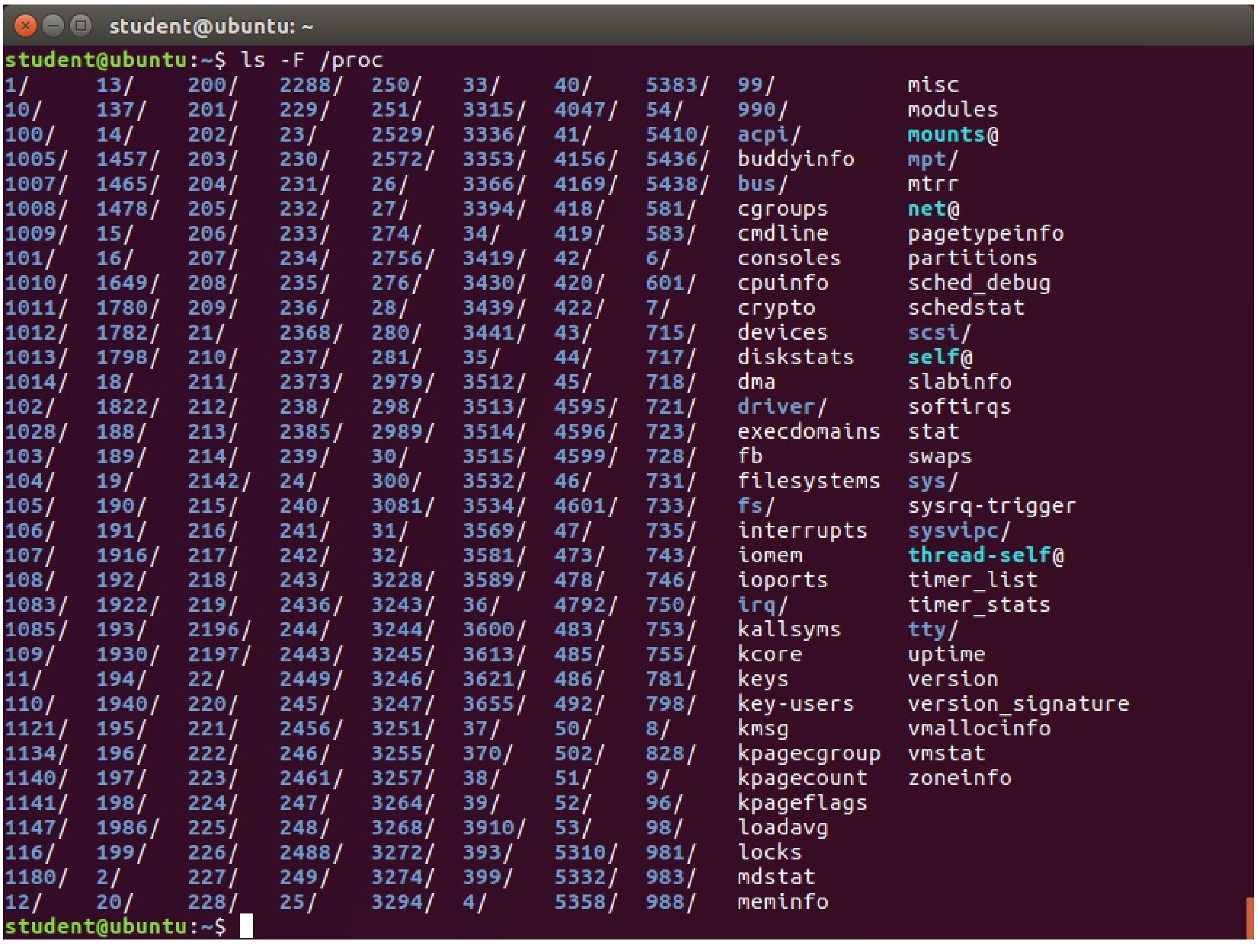
11.8 /proc Basics

`/proc` pseudo-filesystem: long history. Has roots in other UNIX operating system variants. Originally developed to display information about **processes** on system, each of w hich has ow n subdirectory in `/proc` w ith all important process characteristics available.

Over time, grew to contain lot of information about system properties, eg. interrupts, memory, netw orking, etc. in somew hat anarchistic w ay. Still extensively used, w ill often refer to it.

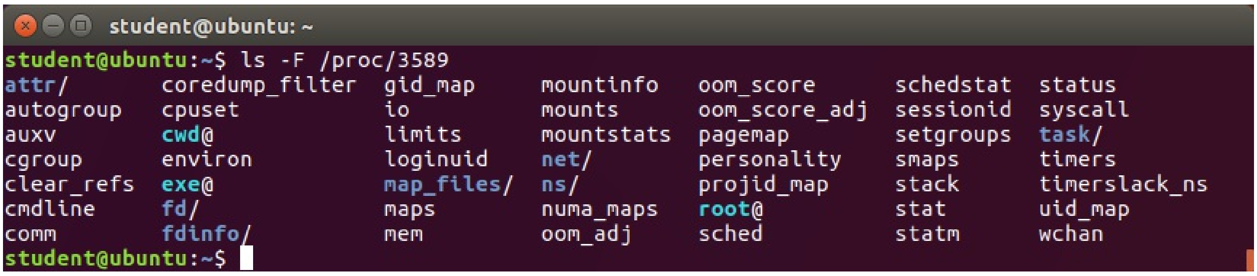
11.9 A survey of /proc

What resides in `/proc` pseudo-filesystem:



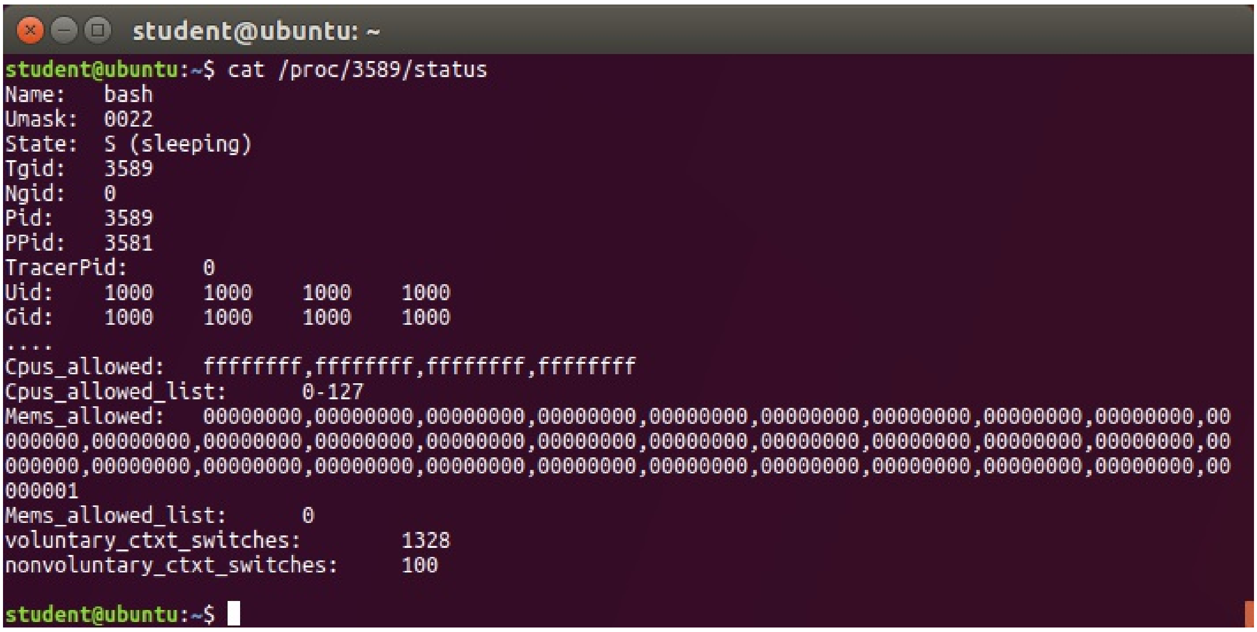
```
student@ubuntu:~$ ls -F /proc
1/     13/    200/   2288/  250/   33/    40/    5383/  99/          misc
10/    137/   201/   229/   251/   3315/  4047/  54/    990/         modules
100/   14/    202/   23/    2529/  3336/  41/    5410/  acpi/        mounts@
1005/  1457/  203/   230/   2572/  3353/  4156/  5436/  buddyinfo    mpt/
1007/  1465/  204/   231/   26/    3366/  4169/  5438/  bus/         mtrr
1008/  1478/  205/   232/   27/    3394/  418/   581/   cgroups      net@
1009/  15/    206/   233/   274/   34/    419/   583/   cmdline      pagetypeinfo
101/   16/    207/   234/   2756/  3419/  42/    6/     consoles     partitions
1010/  1649/  208/   235/   276/   3430/  420/   601/   cpuinfo      sched_debug
1011/  1780/  209/   236/   28/    3439/  422/   7/     crypto       schedstat
1012/  1782/  21/    2368/  280/   3441/  43/    715/   devices      scsi/
1013/  1798/  210/   237/   281/   35/    44/    717/   diskstats    self@
1014/  18/    211/   2373/  2979/  3512/  45/    718/   dma          slabinfo
102/   1822/  212/   238/   298/   3513/  4595/  721/   driver/      softirqs
1028/  188/   213/   2385/  2989/  3514/  4596/  723/   execdomains  stat
103/   189/   214/   239/   30/    3515/  4599/  728/   fb           swaps
104/   19/    2142/  24/    300/   3532/  46/    731/   filesystems  sys/
105/   190/   215/   240/   3081/  3534/  4601/  733/   fs/          sysrq-trigger
106/   191/   216/   241/   31/    3569/  47/    735/   interrupts   sysvipc/
107/   1916/  217/   242/   32/    3581/  473/   743/   iomem        thread-self@
108/   192/   218/   243/   3228/  3589/  478/   746/   ioports      timer_list
1083/  1922/  219/   2436/  3243/  36/    4792/  750/   irq/         timer_stats
1085/  193/   2196/  244/   3244/  3600/  483/   753/   kallsyms     tty/
109/   1930/  2197/  2443/  3245/  3613/  485/   755/   kcore        uptime
11/    194/   22/    2449/  3246/  3621/  486/   781/   keys         version
110/   1940/  220/   245/   3247/  3655/  492/   798/   key-users    version_signature
1121/  195/   221/   2456/  3251/  37/    50/    8/     kmsg         vmallocinfo
1134/  196/   222/   246/   3255/  370/   502/   828/   kpagecgroup  vmstat
1140/  197/   223/   2461/  3257/  38/    51/    9/     kpagecount   zoneinfo
1141/  198/   224/   247/   3264/  39/    52/    96/    kpageflags
1147/  1986/  225/   248/   3268/  3910/  53/    98/    loadavg
116/   199/   226/   2488/  3272/  393/   5310/  981/   locks
1180/  2/     227/   249/   3274/  399/   5332/  983/   mdstat
12/    20/    228/   25/    3294/  4/     5358/  988/   meminfo
student@ubuntu:~$
```

First, see there is subdirectory for each process on system, w hether sleeping, running, scheduled out. Looking at random one:



```
student@ubuntu:~$ ls -F /proc/3589
attr/       coredump_filter  gid_map     mountinfo   oom_score      schedstat  status
autogroup   cpuset           io          mounts      oom_score_adj  sessionid  syscall
auxv        cwd@             limits      mountstats  pagemap        setgroups  task/
cgroup      environ          loginuid    net/        personality    smaps      timers
clear_refs  exe@             map_files/  ns/         projid_map     stack      timerslack_ns
cmdline     fd/              maps        numa_maps   root@          stat       uid_map
comm        fdinfo/          mem         oom_adj     sched          statm      wchan
student@ubuntu:~$
```

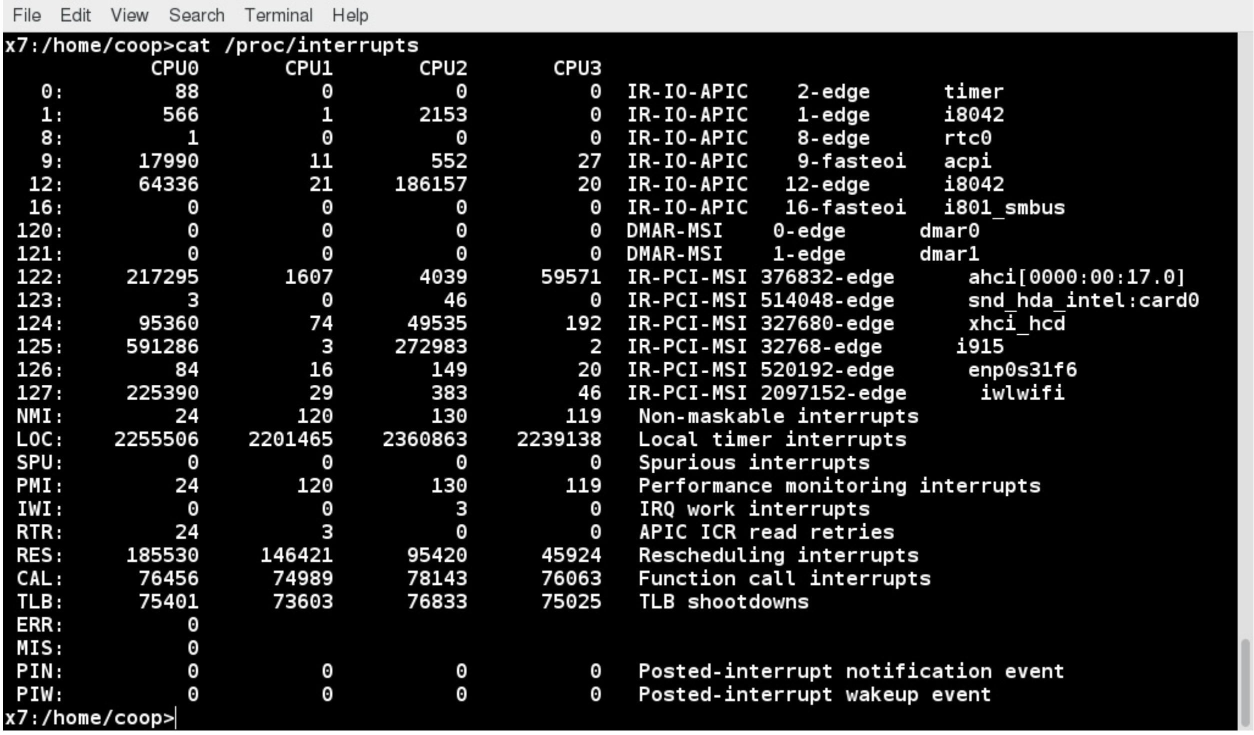
Directory full of information about status of process and resources it is using. For example:



```
student@ubuntu:~$ cat /proc/3589/status
Name:   bash
Umask:  0022
State:  S (sleeping)
Tgid:   3589
Ngid:   0
Pid:    3589
PPid:   3581
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    1000    1000    1000    1000
....
Cpus_allowed:   ffffffff,ffffffff,ffffffff,ffffffff
Cpus_allowed_list:      0-127
Mems_allowed:   00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00
000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00
000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00
000001
Mems_allowed_list:      0
voluntary_ctxt_switches:        1328
nonvoluntary_ctxt_switches:     100

student@ubuntu:~$
```

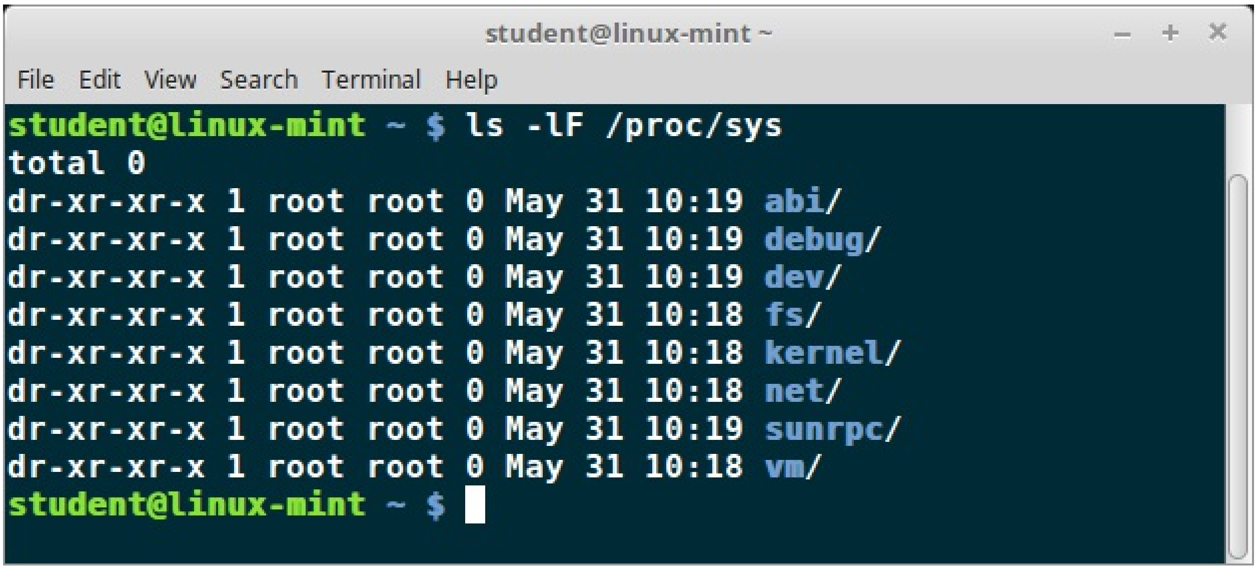
Other entries give system-w ide information. Eg. can see **interrupt** statistics in below screenshot. For each interrupt, see w hat type it is, how many times it has been handled on each CPU, w hich devices registered to respond to it. Also get global statistics.



```
x7:/home/coop>cat /proc/interrupts
            CPU0       CPU1       CPU2       CPU3
   0:         88          0          0          0  IR-IO-APIC    2-edge      timer
   1:        566          1       2153          0  IR-IO-APIC    1-edge      i8042
   8:          1          0          0          0  IR-IO-APIC    8-edge      rtc0
   9:      17990         11        552         27  IR-IO-APIC    9-fasteoi   acpi
  12:      64336         21     186157         20  IR-IO-APIC   12-edge      i8042
  16:          0          0          0          0  IR-IO-APIC   16-fasteoi   i801_smbus
 120:          0          0          0          0  DMAR-MSI    0-edge      dmar0
 121:          0          0          0          0  DMAR-MSI    1-edge      dmar1
 122:     217295       1607       4039      59571  IR-PCI-MSI 376832-edge      ahci[0000:00:17.0]
 123:          3          0         46          0  IR-PCI-MSI 514048-edge      snd_hda_intel:card0
 124:      95360         74      49535        192  IR-PCI-MSI 327680-edge      xhci_hcd
 125:     591286          3     272983          2  IR-PCI-MSI 32768-edge      i915
 126:         84         16        149         20  IR-PCI-MSI 520192-edge      enp0s31f6
 127:     225390         29        383         46  IR-PCI-MSI 2097152-edge      iwlwifi
 NMI:         24        120        130        119   Non-maskable interrupts
 LOC:    2255506    2201465    2360863    2239138   Local timer interrupts
 SPU:          0          0          0          0   Spurious interrupts
 PMI:         24        120        130        119   Performance monitoring interrupts
 IWI:          0          0          3          0   IRQ work interrupts
 RTR:         24          3          0          0   APIC ICR read retries
 RES:     185530     146421      95420      45924   Rescheduling interrupts
 CAL:      76456      74989      78143      76063   Function call interrupts
 TLB:      75401      73603      76833      75025   TLB shootdowns
 ERR:          0
 MIS:          0
 PIN:          0          0          0          0   Posted-interrupt notification event
 PIW:          0          0          0          0   Posted-interrupt wakeup event
x7:/home/coop>
```

11.10 /proc/sys

Most tunable system parameters can be found in subdirectory tree rooted at `/proc/sys` :



```
student@linux-mint ~ $ ls -lF /proc/sys
total 0
dr-xr-xr-x 1 root root 0 May 31 10:19 abi/
dr-xr-xr-x 1 root root 0 May 31 10:19 debug/
dr-xr-xr-x 1 root root 0 May 31 10:19 dev/
dr-xr-xr-x 1 root root 0 May 31 10:18 fs/
dr-xr-xr-x 1 root root 0 May 31 10:18 kernel/
dr-xr-xr-x 1 root root 0 May 31 10:18 net/
dr-xr-xr-x 1 root root 0 May 31 10:19 sunrpc/
dr-xr-xr-x 1 root root 0 May 31 10:18 vm/
student@linux-mint ~ $
```

Each subdirectory contains information + knobs that can be tuned (w ith care):

- `abi/` : Contains files w ith application binary information; rarely used
- `debug/` : Debugging parameters; for now, just some control of exception reporting
- `dev/` : Device parameters, including subdirectories for **cdrom**, **scsi**, **raid**, **parport**
- `fs/` : Filesystem parameters, including quote, files handles used, and maximums, inode and directory information, etc.
- `kernel/` : Kernel parameters. Many important entries here.
- `net/` : Netw ork parameters. Subdirectories for **ipv4**, **netfilter**, etc.
- `vm/` : Virtual memory parameters. Many important entries here.

View ing/changing parameters can be done w ith simple commands. Eg. maximum number of threads allow ed on system seen by looking at:

```
$ ls -l /proc/sys/kernel/threads-max
$ cat /proc/sys/kernel/threads-max
129498
```

Can then modify value, verify change w as effected:

```
$ sudo bash -c 'echo 100000 > /proc/sys/kernel/threads-max'
$ cat /proc/sys/kernel/threads-max
100000
```

Remember from discussion of **sysctl**, same effect accomplished by:

```
$ sudo sysctl kernel.threads-max=100000
```

View ing value can be done as normal user, changing requires superuser privilege.

11.11 /sys Basics

`/sys` pseudo-filesystem: integral part of **Unified Device Model**. Conceptually, based on **device tree**, one can w alk through it and see buses, devices, etc. Also now contains information w hich may or may not be strictly related to devices, such as kernel modules.

Has more tightly defined structure than `/proc`. Most entries contain only one line of text (although there are exceptions) unlike precursor which has many multi-line entries whose exact contents may change between kernel versions. Thus, interface hopefully more stable.

There are system properties which have display entries in both `/proc` and `/sys`. For compatibility with widely used system utilities, older forms only gradually being whittled down.

11.12 A Survey of /sys

Support for **sysfs** virtual filesystem built into all modern kernels, should be mounted under `/sys`. However, unified device model does not require mounting **sysfs** in order to function.

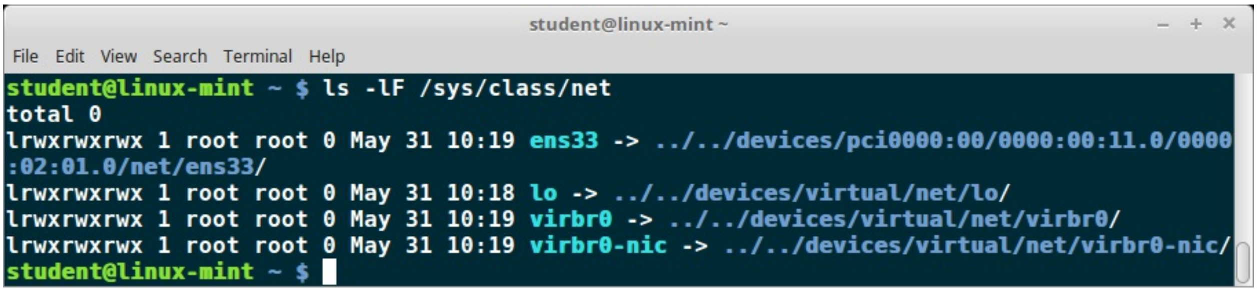
Taking look at 3.18 kernel (warning; exact layout of this filesystem tends to mutate). Top level directory command yields:

```
$ ls -F /sys
block/ bus/ class/ dev/ devices/ firmware/ fs/ kernel/ module/ power/
```

which displays basic device hierarchy. Device model **sysfs** implementation also includes information not strictly related to hardware.

Network devices examined with:

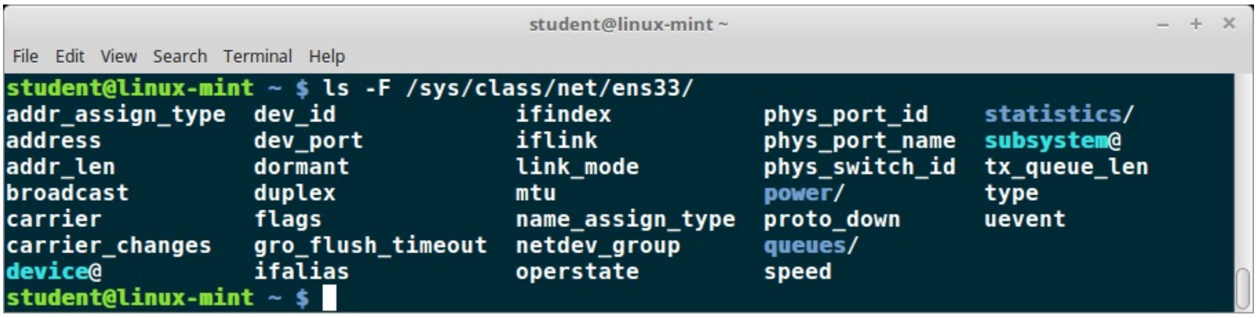
```
$ ls -lF /sys/class/net
```



```
student@linux-mint ~ $ ls -lF /sys/class/net
total 0
lrwxrwxrwx 1 root root 0 May 31 10:19 ens33 -> ../../devices/pci0000:00/0000:00:11.0/0000
:02:01.0/net/ens33/
lrwxrwxrwx 1 root root 0 May 31 10:18 lo -> ../../devices/virtual/net/lo/
lrwxrwxrwx 1 root root 0 May 31 10:19 virbr0 -> ../../devices/virtual/net/virbr0/
lrwxrwxrwx 1 root root 0 May 31 10:19 virbr0-nic -> ../../devices/virtual/net/virbr0-nic/
student@linux-mint ~ $
```

Below, can see what looking at Ethernet card gives.

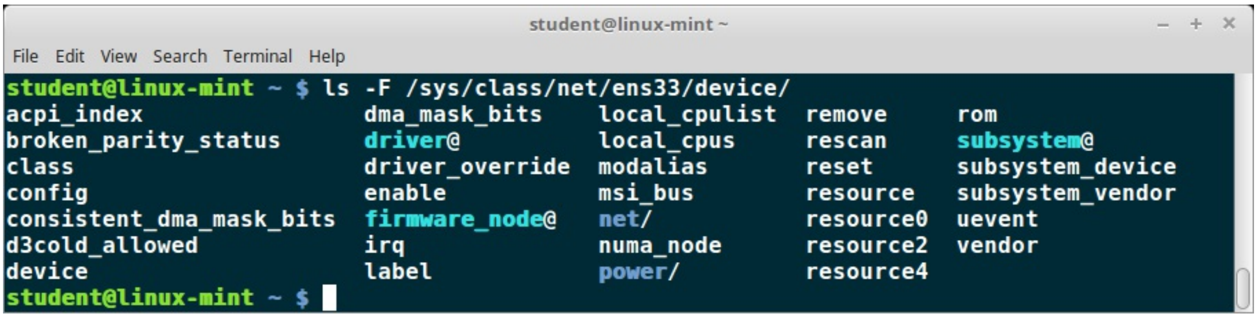
Intention with **sysfs** to have one text value per line, although not expected to be rigorously enforced.



```
student@linux-mint ~ $ ls -F /sys/class/net/ens33/
addr_assign_type  dev_id             ifindex           phys_port_id     statistics/
address           dev_port           iflink            phys_port_name   subsystem@
addr_len          dormant            link_mode         phys_switch_id   tx_queue_len
broadcast         duplex             mtu               power/           type
carrier           flags              name_assign_type  proto_down       uevent
carrier_changes   gro_flush_timeout  netdev_group      queues/
device@           ifalias            operstate         speed
student@linux-mint ~ $
```

Underlying device and driver for first network interface can be traced through `device` and (to be seen shortly) `driver` symbolic links. Below shows what can be seen when looking at directory corresponding to first Ethernet card.

To see full spectrum of information available with **sysfs**, will just have to examine.



```
student@linux-mint ~ $ ls -F /sys/class/net/ens33/device/
acpi_index              dma_mask_bits    local_cpulist  remove     rom
broken_parity_status    driver@          local_cpus     rescan     subsystem@
class                   driver_override  modalias       reset      subsystem_device
config                  enable           msi_bus        resource   subsystem_vendor
consistent_dma_mask_bits firmware_node@  net/           resource0  uevent
d3cold_allowed          irq              numa_node      resource2  vendor
device                  label            power/         resource4
student@linux-mint ~ $
```

11.13 sar

sar: **S**ystems **A**ctivity **R**eporter. All-purpose tool for gathering system activity + performance data, creating reports readable by humans.

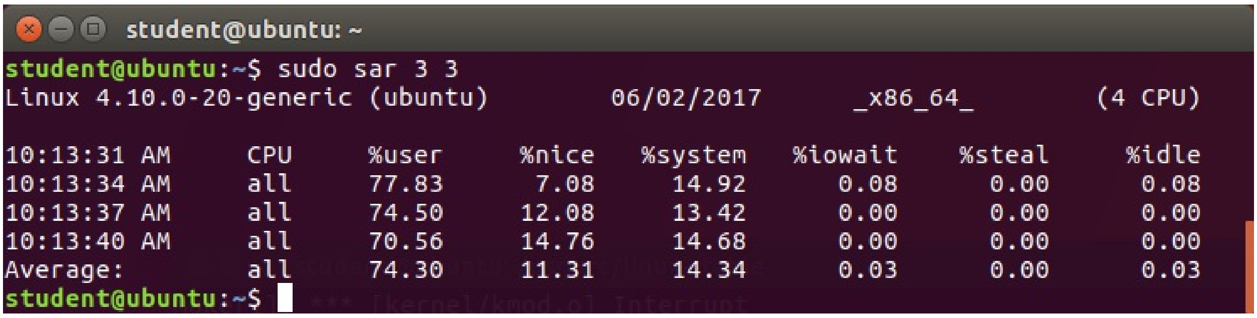
On Linux systems, backend to **sar** is **sadc** (system activity data collector) w hich actually accumulates statistics. Stores information in `/var/log/sa` directory, w ith daily frequency by default, but w hich can be adjusted. Data collection can be started from command line, regular periodic collection usually started as **cron** job stored in `/etc/cron.d/sysstat`.

sar then reads in this data (either from default locations or by use of file specified w ith `-f` option), then produces report.

sar invoked via:

```
$ sar [ options ] [ interval ] [ count ]
```

w here report repeated after interval seconds a total of count times (defaults to 1). With no options, gives report on CPU usage.



```
student@ubuntu:~$ sudo sar 3 3
Linux 4.10.0-20-generic (ubuntu)        06/02/2017      _x86_64_        (4 CPU)

10:13:31 AM     CPU     %user     %nice   %system   %iowait    %steal     %idle
10:13:34 AM     all     77.83      7.08     14.92      0.08      0.00      0.08
10:13:37 AM     all     74.50     12.08     13.42      0.00      0.00      0.00
10:13:40 AM     all     70.56     14.76     14.68      0.00      0.00      0.00
Average:        all     74.30     11.31     14.34      0.03      0.00      0.03
student@ubuntu:~$
```

List of major **sar** options, or modes, each one of w hich has its ow n sub-options:

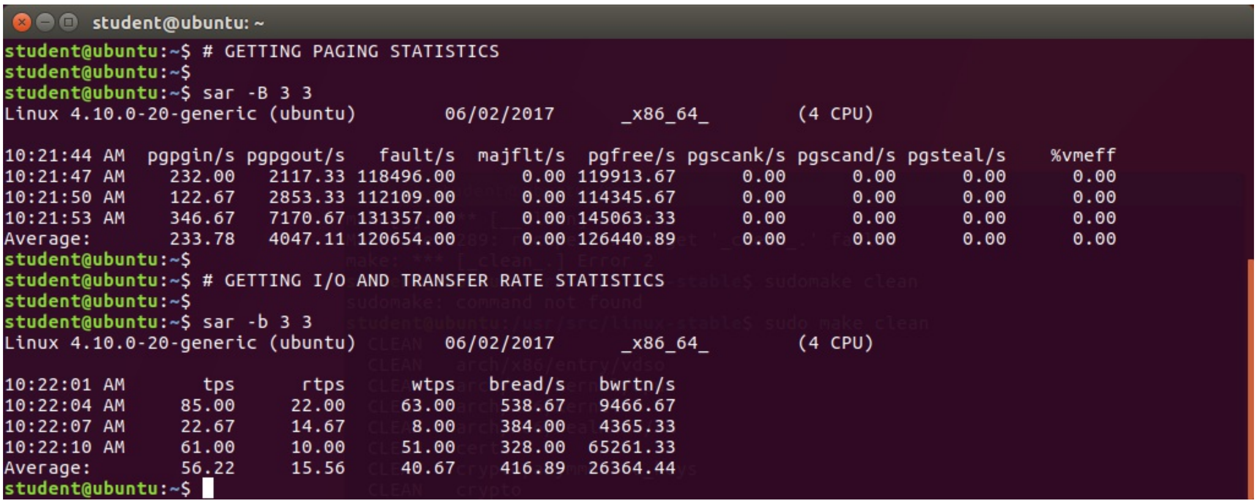
sar Options

Option	Meaning
`-A`	Almost all Information
`-b`	I/O and transfer rate statistics (similar to **iostat**)
`-B`	Paging statistics including page faults
`-x`	Block device activity (similar to **iostat -x**)
`-n`	Netw ork statistics
`-P`	Per CPU statistics (as in `sar -P ALL 3`)

`-q`	Queue lengths (run queue, processes, and threads)
`-r`	Swap and memory utilization statistics
`-R`	Memory statistics
`-u`	CPU utilization (default)
`-v`	Statistics about inodes and files and files handles
`-w`	Context switching statistics
`-W`	Swapping statistics, pages in and out per second
`-f`	Extract information from specified file, created by the `-o` option
`-o`	Save readings in the file specified, to be read in later with `-f` option

For example, below can take look at getting paging statistics, and then I/O and transfer rate statistics.

ksar program -> **java**-based utility for generating nice graphs for **sar** data. Can be downloaded from https://sourceforge.net/projects/ksar/.



```
student@ubuntu:~$ # GETTING PAGING STATISTICS
student@ubuntu:~$
student@ubuntu:~$ sar -B 3 3
Linux 4.10.0-20-generic (ubuntu)        06/02/2017      _x86_64_        (4 CPU)

10:21:44 AM  pgpgin/s pgpgout/s   fault/s  majflt/s  pgfree/s pgscank/s pgscand/s pgsteal/s    %vmeff
10:21:47 AM    232.00   2117.33 118496.00      0.00 119913.67      0.00      0.00      0.00      0.00
10:21:50 AM    122.67   2853.33 112109.00      0.00 114345.67      0.00      0.00      0.00      0.00
10:21:53 AM    346.67   7170.67 131357.00      0.00 145063.33      0.00      0.00      0.00      0.00
Average:       233.78   4047.11 120654.00      0.00 126440.89      0.00      0.00      0.00      0.00
student@ubuntu:~$
student@ubuntu:~$ # GETTING I/O AND TRANSFER RATE STATISTICS
student@ubuntu:~$
student@ubuntu:~$ sar -b 3 3
Linux 4.10.0-20-generic (ubuntu)        06/02/2017      _x86_64_        (4 CPU)

10:22:01 AM       tps      rtps      wtps   bread/s   bwrtn/s
10:22:04 AM     85.00     22.00     63.00    538.67   9466.67
10:22:07 AM     22.67     14.67      8.00    384.00   4365.33
10:22:10 AM     61.00     10.00     51.00    328.00  65261.33
Average:        56.22     15.56     40.67    416.89  26364.44
student@ubuntu:~$
```

##

Back to top

Previous Chapter - Table of Contents - Next Chapter