# *Basic Hardware Utilities*

**lspci** Display info about all the PCI buses and all the peripheral components connected to a computer.
**-k** shows the kernel drivers that support the device. **-v** for verbose on most of these ls commands
**-t** displays a tree diagram that shows connections between all busses, bridges, and devices.
**lsusb** Display all the USB components connected to a computer. gives more info
**-v** for verbose, **-s bus_name** to specify a bus
There is also **lshw, lshal**, **lscpu**, **lsblk**, **lspcmcia**
**lshw -X -** a graphic frontend
**lshw -html > hardware.html** -- Create a html overview of the hardware

**hwinfo --<hw_item>** - displays HW info using libhd **- <hw_item>** is one of the following: all, bios, block, bluetooth, braille, bridge, camera, cdrom, chipcard, cpu, disk, dsl, dvb, fingerprint, floppy, framebuffer, gfxcard, hub, ide, isapnp, isdn, joystick, keyboard, memory, modem, monitor, mouse, netcard, network, partition, pci, pcmcia, pcmcia-ctrl, pppoe, printer, scanner, scsi, smp, sound, storage-ctrl, sys, tape, tv, usb, usb-ctrl, vbe, wlan, zip
--short - for brief output
--debug level - set debuglevel
**hwinfo --disk** show all disks
**hwinfo --short --block** just an overview
**hwinfo --disk --only /dev/sdb** - show a particular disk
**hwinfo --disk --save-config=all** - save disk config state
**hwprobe=bios.ddc.ports=4 hwinfo --monitor** - try 4 graphics card ports for monitor data (default: 3)

**kudzu [options]**
- deprecated in RHEL6 - probed and compared with /etc/sysconfig/hwconfig. Configuration options, ran at boot time.

**Kernel State Monitoring Utilities**
**uname** OS name, version, license, processor, and hardware details.
**uptime** Duration the system has been running, number of users currently logged on, and load averages
**tload** Graphical representation of load average for the past 1 minute, 5 minutes, and 15 minutes.
**w** is essentially a combination of uptime, who, and ps -a

**Memory Monitoring Utilities**
**free** Total memory: free, used, shared, buffered, and cached. -m for megabytes, -k for kb, -s delay
**vmstat** Virtual memory usage; I/O address info, processor allocation currently running
**pmap** Display the mapping of processes with memory resources.
**iostat** Reports on CPU and device utilization. Usage statistics for storage devices and partitions.
**memstat -a**
**systat -m**

**Disk and Filesystem Utilities**
**partprobe** - tells the kernel to re-read the partition table
**-d** Cancel any updates; **-s** Display the storage devices and their partitions.
**df -h** list volumes- memnonic disk free
**du -h -d1** file sizes, human readable, depth=1 --max-depth=1; **-c** shows combined total at bottom
**cifsiostat** - Average reads and writes/sec vs those ops issued per sec, files opened/closed/deleted per sec

**Other useful stuff:**
**id** - lists current user's uid, gid, and group memberships
**file <filename>** Dumps file info including file type
**stat <filename.txt>** Shows full attributes of file
**which <command> -a** Shows the pathnames for the command using $path env var
**lsof | grep <keyword>** List open files, filter by keyword
Remember you can use the watch command (like **watch free**) to have it update on the screen
**watch -n 1 -d free** Sets a 1 sec interval and -d will highlight values that changed

*Make a baseline of system performance on a newly installed Linux system*
**/proc/meminfo** - the used and unused memory and the shared memory and buffers used by the kernel.
**/proc/cpuinfo** - the CPU information and system architecture dependent items
**dmesg** displays a snapshot of information about the hardware that is controlled by the kernel, and that output can be redirected to a file for use in system baseline documentation.

# *Process Management*

**Moving a running job from foreground to background**
- Typing ctrl-z *stops* and places in the background, then type **bg** to *resume* in background
- Type **jobs -l** to see it listed as running (lists all background jobs)
- Executing **fg <job#>** with the number it is listed as will bring it to the foreground
- **dd if=/dev/zero of=/dev/null &** - Copying nothing to nowhere;
- The **&** sends it to the background upon execution without having to stop and resume

[A "shell job" is any command run from the shell. Processes belong to the shell from which they were started]
**nohup ./test > test.output** - makes process keep running when terminal/tty has been closed
(means to run without attaching process to parent terminal)

Kill Signals
**kill -9 <PID>** - brute-force terminate process
**killall <program-name>** - affects all matching processes running

| Num. | Name | Default Action | Description |
|---|---|---|---|
| 1 | SIGHUP | terminate process | Terminal line hangup- restarts the process. After doing so, the process will keep the same PID that it had before. Useful for restarting a service after making changes to a configuration file. |
| 2 | SIGINT | terminate process | Interrupt program - sends a ctrl-c key sequence |
| 3 | SIGQUIT | create core image | Quit program |
| 6 | SIGABRT | create core image | Abort program (formerly SIGIOT) |
| 9 | SIGKILL | terminate process | Brute-force kill; process may not clean up - resources allocated may remain allocated until the system is restarted |
| 15 | SIGTERM | terminate process | Default- allows process to clean up before exiting |

***CPU Priority***
"Niceness" values for CPU priority - remember that -20 is highest priority and 19 is the lowest. Default is 0
- Use **nice** for *starting* programs: **nice -19 ./test** (or PID) - sets to 19 (is not minus 19, which would be set with **--19**)
- Use **renice** for adjusting currently running processes: **renice <priority> -user** (or PID)
- **renice 17 -p 1234** changes the nice value of the job with process id 1134 to 17; no dash for command option
- Change the nice value of process 1234 to -3 with: **renice -3 -p 1234**
- Modify the priority of all processes that belong to a group or user with the -g and -u (instead of -p for process)
- It is recommended changing nice in steps of -5 at a time
- *Only root can apply negative nice values*
- You can set the default nice value of a particular user or group in the /etc/security/limits.conf
- It uses this syntax: [ *username* ] [ hard | soft ] priority [ *nice value* ]
  user12 hard priority 1

The **-n** modifier produces different behavior in nice and renice
- In nice, using -n option adjusts the value from the default (0) and the "-" will actually work as minus.
- Increase the priority: **nice -n -5 ./test.sh** - Decrease the priority: **nice -n 5 ./test.sh**
- So using the -n makes it less confusing with the whole issue of whether it is a minus or a simple hyphen
- In renice, the -n option specifies the actual number so **renice -n -19 -p 3534** sets the nice value to -19

If you have to mess with nice alot, then it is a sign to increase system resources (RAM, but primarily CPU) to handle the load on the server)

# *Using PS to Inspect Processes*

The ps command 3 syntax styles. BSD style - options are not preceded with a dash (**ps aux**); UNIX/LINUX style - options are preceded by a dash (**ps -ef**) "ps aux" is not the same as "ps -aux". "-u" is used to show process of that user, but plain "u" means show detailed information in the other mode. Usually these can be mixed in Linux: For example "ps ax -f". Finally, some options use the GNU style with double-dashes preceding a word, like --forest.

Most often used is display all processes. The "u" or "-f" options display more detailed info

**$ ps ax** --or-- **$ ps aux**
**$ ps -ef** --or-- **$ ps -ef -f** *Using -F can sometimes be more effective than -f*

*Other options (there are many):*

| | |
|---|---|
| **-u <username>** | Display process by user -Multiple usernames can be provided separated by a comma |
| **-f --or-- -F** | Show detailed info |
| **-C <processname>** | Search by name (must be exact) |
| **-p** | Search by PID (separate with comma for multiple e.g., **$ ps -f -p 3150,7298,6544** |
| **--sort=** | Prefix with a "-" or "+" to sort descending/ascending (e.g., **$ ps aux --sort=-pcpu,+pmem**) |
| **--forest** | Displays ASCII art of process tree. Sort and forest don't work well together |
| **--ppid <PID>** | Search by parent PID (show child processes) |
| **-o** | Show listed columns sep with commas (e.g., **$ ps -e -o pid,uname,pcpu,pmem**) |
| **-L** | Show threads of a process, (e.g., **$ ps -p 3150 -L or ps -C firefox-bin -L -o pcpu,state**) |

- Piping to grep instead of using the -C option can be more convenient **$ ps -ef | grep apache**
- Fields for --sort and -o options are available in the manpage section "Standard Format Specifiers" (there are tons)
- Rename the column labels using the format "-o pid,uname=USERNAME,pcpu=CPU_USAGE,pmem,comm"
- When listing usernames, if the length is greater than 8 characters then ps will show the UID instead of username.
- As usual pipe to "less" to ease reading. If sorting remember to consider piping to head 5 or tail -5 as needed, etc.

**Common attributes to query/ sort by, etc.**

The man page lists many attributes you can interrogate using ps. Here are some of the commonly used ones.

pid, comm (just the command name), cmd (the command with all its arguments as a string), uname (username running process), pmem (percent of physical memory being used), pcpu (percent of CPU being used), nice (the niceness value), etime (elapsed time running), state (one letter state code), stat (multiple character state code listing)

```
[root@localhost ~]# ps afx
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 init [5]
    2 ?        S<     0:00 [migration/0]
 2438 tty5     Ss+    0:00 /sbin/mingetty tty5
 2439 tty6     Ss+    0:00 /sbin/mingetty tty6
 2440 ?        Ss     0:00 /usr/sbin/gdm-binary -nodaemon
 2506 ?        S      0:00  \_ /usr/sbin/gdm-binary -nodaemon
 2511 tty7     Ss+    0:21      \_ /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth -nolist
 2525 ?        Ss     0:00      \_ /usr/bin/gnome-session
 2561 ?        Ss     0:00          \_ /usr/bin/ssh-agent /bin/sh -c exec -l /bin/bash -c "/usr/
```

The "f" option shows child process as shown above

```
[root@localhost ~]# ps alx
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4     0     1     0  15   0   2072   588 -      Ss   ?          0:00 init [5]
1     0     2     1 -100  -      0     0 migrat S<   ?          0:00 [migration/0]
1     0     3     1  34  19      0     0 ksofti SN   ?          0:00 [ksoftirqd/0]
```

The "l" option lists more info

```
[root@localhost ~]# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2072   588 ?        Ss   13:40   0:00 init [5]
root         2  0.0  0.0      0     0 ?        S<   13:40   0:00 [migration/0]
root         3  0.0  0.0      0     0 ?        SN   13:40   0:00 [ksoftirqd/0]
```

The "x" option shows info on CPU and RAM usage

**PS output columns:**

VSZ= virtual memory size, RSS= resident memory size, STAT= state (S=sleep, R=running, etc), F represents number of forked processes, PPID is parent PID. WCHAN is the name of the kernel function the process is sleeping in

***States:***

I - Idle (sleeping >20 seconds).
R - Running or runnable (on run queue)
S - Sleeping <20 seconds. (waiting for an event to complete)
T - A stopped process (either by a job control signal or because it is being traced)
D - Uninterruptible sleep (usually IO)
U - Process in uninterruptible wait.
Z - Marks a dead ("zombie") process
W - Paging (not valid since the 2.6.xx kernel)
X - Dead (should never be seen)

*Additional state codes*

+ - Is in the foreground process group of its control terminal.
< - Has raised CPU scheduling priority.
N - Has reduced CPU scheduling priority (see setpriority(2)).
> - Has specified a soft limit on memory requirements and is exceeding that limit; such a process is not swapped.
A - Process has asked for random page replacement
E - Process is trying to exit.
l - Is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
L - Has pages locked in memory (for example, for raw I/O).
s - Is a session leader.

***Some examples***

| | |
|---|---|
| Display top 5 CPU using processes. | **ps aux --sort=-pcpu \| head -5** |
| Display elapsed time of processes | **ps -e -o pid,comm,etime** |
| View process in realtime | **watch -n 1 'ps -e -o pid,pmem,pcpu --sort=-pmem \| head -15'** |
| Output PID, command, and nice value | **ps -o pid,comm,nice -p 594** |
| List by % cpu with sed | **ps -e -o pcpu,cpu,nice,cputime,args –sort pcpu \| sed '/^ 0.0 /d'** |
| List by mem (KB) usage | **ps -e -orss=,args= \| sort -b -k1,1n \| pr -TW$COLUMNS** |
| Display process hierarchy in a tree style | **ps -f --forest -C apache2** |

Display child processes of a parent process

**$ ps -o pid,uname,comm -C apache2**

| PID | USER | COMMAND |
|---|---|---|
| 2359 | root | apache2 |
| 4524 | www-data | apache2 |
| 4525 | www-data | apache2 |

The first process owned by root is the main apache2 process and all other apache2 processes have been forked out of this. We now interrogate that here:

**$ ps --ppid 2359**

| PID | TTY | TIME | CMD |
|---|---|---|---|
| 4524 | ? | 00:00:00 | apache2 |
| 4525 | ? | 00:00:00 | apache2 |
| **...** | | | |

**pstree** - outputs a tree view of processes, which can be preferable over **ps --forest**

```
[root@localhost ~]# pstree
init─┬─acpid
     ├─atd
     ├─auditd─┬─audispd───{audispd}
     │        └─{auditd}
     ├─automount───4*[{automount}]
```

# *TOP*

Combines **ps**, **uptime**, **free** and updates regularly; default sorts task by CPU usage- can kill(k), renice(r)
To hide / show header lines and increase display space, press 'l' for load avg (first line); pressing 't' toggles CPU states (2nd/ 3rd lines); and pressing 'm' toggles memory info (4th/ 5th lines)

**The typical header:**

```
top - 07:18:59 up 17:38,  2 users,  load average: 0.21, 0.15, 0.10
Tasks: 116 total,   3 running, 113 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  0.0%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1035108k total,  1014336k used,    20772k free,   137732k buffers
Swap:  2097144k total,        0k used,  2097144k free,   646340k cached
```

- First line displays same contents as **uptime** with CPU load average provided in 1, 5, and 15 minute readings
- Numbers are "by average how many runnable processes are in the runqueue waiting to be served by the kernel's scheduler" meaning one-at-a-time per cpu core.
- What is the "optimal setting?" If you have one CPU, the optimal setting would be 1, 16 cpus, 16
- Press "1" to list CPU lines by individual CPUs instead of a total (this is the third line down that says %CPU(#)

The CPU line displays the following:

**us** - user space
**sy** - system - kernel space
**ni** - user nice - time spent on low priority processes
**id** - idle - time spent idle
**wa** - I/O wait cpu time - time spent in wait (on disk)
**hi** - hardware irq - time spent servicing/handling hardware interrupts
**si** - software irq - time spent servicing/handling software interrupts
**st** - steal time - involuntary wait by virtual CPU while a hypervisor is servicing another processor (or) %CPU time stolen from a virtual machine

The memory line is identical to running the **free** command
- About 30% memory should always be available for buffer and cache. If you don't have that, or are loading too many programs, the least active programs will release the pages they have allocated to free up memory
- Cache - structured data: filenames, etc - Buffer- unstructured data like parts of files to write to disk
- Swap is allows cache and buffers dedicated to RAM for operations

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      15   0  2072  596  512 S  0.0  0.1   0:00.36 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.51 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
```

By default, processes are listed by CPU usage, < and > sort by the relatively adjacent column in that direction.
Press 'M' key to sort the process list by memory usage
Press 'P' - to sort the process list by CPU usage.
Press 'N' - to sort the list by process id
Press 'T' - to sort by the running time.
Press 'R' - to sort in reverse order
Type F/f for available fields. Added fields have to be saved to use later by typing 'W' (to write config to **~/.toprc**)
Update the output on-demand by pressing the space bar.
Typing 'x' will boldface the column the processes are sorted by. Use b for a bg color instead
c - will show or hide the command's absolute path, and arguments
n - change the number of processes to display (0 for maximum)
d - change output update frequency, will prompt to enter new time in seconds
O/o - to change the order the columns are displayed (left-to-right)
h or ?- display help for interactive top commands
1 - display all CPUs/ cores separately
A - Split output into 4 individually configurable/sortable views; then use "a" to cycle between different views.
b or z - highlight running process
Type 'k' to kill and send specific signal to process (default is 15 SIGTERM), 'r' to renice; 'S' to just send signal
U/u - Filter processes of a specific user
Typing V gives a 'forest' view similar to the same usage in ps

Typing 'F' brings up this window allowing to choose a column to sort processes by

```
Current Sort Field:  K  for window 3:Mem
Select sort field via field letter, type any other key to return
  a: PID        = Process Id                 u: nFLT       = Page Fault count
  b: PPID       = Parent Process Pid         v: nDRT       = Dirty Pages count
  c: RUSER      = Real user name             w: S          = Process Status
  d: UID        = User Id                    x: COMMAND    = Command name/line
  e: USER       = User Name                  y: WCHAN      = Sleeping in Function
  f: GROUP      = Group Name                 z: Flags      = Task Flags <sched.h>
  g: TTY        = Controlling Tty
  h: PR         = Priority                 Note1:
  i: NI         = Nice value                 If a selected sort field can't be
  j: P          = Last used cpu (SMP)        shown due to screen width or your
* K: %CPU       = CPU usage                  field order, the '<' and '>' keys
  l: TIME       = CPU Time                   will be unavailable until a field
  m: TIME+      = CPU Time, hundredths       within viewable range is chosen.
  n: %MEM       = Memory usage (RES)
  o: VIRT       = Virtual Image (kb)       Note2:
  p: SWAP       = Swapped size (kb)          Field sorting uses internal values,
  q: RES        = Resident size (kb)         not those in column display.  Thus,
  r: CODE       = Code size (kb)             the TTY & WCHAN fields will violate
  s: DATA       = Data+Stack size (kb)       strict ASCII collating sequence.
  t: SHR        = Shared Mem size (kb)       (shame on you if WCHAN is chosen)
```

Different ways of starting up top:
**top -u <username>** - display process from specific user
**top -p 1208, 1425** - to display specific PIDs
**top -n 1 -b > top-output.txt** --use 'batch' output to a file or pipe (text rather than binary) -n is iterations
**top -i** --to display only active processes (ignore idle)

Be aware that the top command comes in various variants and each has a slightly different set of options and method of usage. To check your top command version and variant use the -v option
For example, in some versions of top, o/O provides a filter prompt to search processes by various criteria
*HTOP offers more ease of use but uses 5x as many system resources*

When reading the virt / resident memory columns in top, remember it is showing pages- memory pages by default are about 4k so multiply what you see by 4 to get actual memory size (resident memory is the physical memory)

## *How Much RAM is Really Free? (disk buffers/cache)*

Remember that some RAM is available for applications, and simultaneously used for disk buffer/ cached

**$ free -m**

| | total | used | free | shared | buffers | cached |
|---|---|---|---|---|---|---|
| Mem: | 1504 | 1491 | 13 | 0 | 91 | 764 |
| -/+ buffers/cache: | | 635 | 869 | | | |
| Swap: | 2047 | 6 | 2041 | | | |

If you don't know how to read the numbers, you'll think the ram is 99% full when it's really just 42% (635MB). Notice also 635+869-13=1491 "used" and 91+764+13=~869 "free"

**$ free -m**

| | total | used | free | shared | buffers | cached |
|---|---|---|---|---|---|---|
| Mem: | 1834 | 1757 | 76 | 9 | 0 | 1021 |
| -/+ buffers/cache: | | 735 | 1098 | | | |
| Swap: | 2047 | 0 | 2047 | | | |

Here 735MB used.
735+1098-76=1757 called "used" - (used+free in buffers/cache minus "free" in mem = reported used in mem).
Shared is usually shared libs. Buffers is when data needs to be committed to disk
Cache is big because files moved to RAM (especially frequently used files) are kept in RAM as long as possible (optimization). Cache is generally freed up when resources are needed for reallocation.
Also consider 1757 "used"-1021 cached is = 735 really used. (and 735+1098 is 1834 total reported)
Here we really have 1098 available

*The -/+ buffers/cache numbers are the real deal.*

# *The sysstat Suite and sar*

The sysstat suite includes the programs **iostat** (CPU, I/O stats), **mpstat** (CPU stats), **pidstat** (stats on processes), **nfsiostat** (NFS-related stats), and **cifsiostat** (CIFS stats). The flagship of the suite is the **sar** collection and reporting suite, described below. The other tools have their own expanded descriptions afterward

**sadc - System Activity Data Collector**. writes binary data to /var/log/sa/ named "sa<dd>" (d is day of month)
**sa1**- script to run sadc with cron or systemd. Collects and stores binary data in the system activity daily data file
**sar - System Activity Reporter** produces reports from sadc files. Writes ascii files to /var/log/sa/ named "sar<dd>"
**sa2** - script to run sar with cron or systemd. Writes a summarized daily activity report
**sadf** outputs sar reports in CSV, XML, JSON, etc. so you can output to other programs. Can also SVG graphs.

**Installation**
- If installing from source: **./configure --enable-install-cron**, then **make**, then **make install**
- The option **--enable-install-cron** makes sure to creates **/etc/rc.d/*** stuff for you
- Installs executables under /usr/local/bin
- (old issue?) Run **echo alias sar='LANG=C sar' >> /etc/bashrc** to ensure sar uses 24 hour clock

**Ensuring cron is set up**
If your version of sysstat didn't do it, create a file under /etc/cron.d directory that will collect the historical sar data.
*vi /etc/cron.d/sysstat*
*/10 * * * * root /usr/local/lib/sa/sa1 1 1
53 23 * * * root /usr/local/lib/sa/sa2 -A

- If installed from source, sa1 and sa2 is probably in **/usr/local/lib/sa**
- If installed with a package manager, they may be located in be **/usr/lib/sa/sa1** and **/usr/lib/sa/sa2**
- The above runs sa1 to collect data every 10 minutes, and compile that data about midnight with sa2
- sa1 is executed with options "1 1" - run once with 1 second interval

**Log management**
Generated log files are only kept for a limited time (usually ~28 days). One option is to have a script back them up before deletion- however, the configuration file /etc/sysconfig/sysstat holds a HISTORY parameter to the number of days you want to keep the log files. The maximum is supposedly 28 days, but if set to more, log files will be stored in a month-by-month directory. Then, log files will be pointing to a symlink such as YYYYMM/saDD

**$ sar -u ALL 1 3** - report today's CPU usage (so far) every 1 sec, 3 times; the ALL option gives extended info
**$ sar -u -f /var/log/sa/sa10** - Specify reports for a specific day (file); here the 10th day of this month (in the file sa10)
**$ sar -P 1** - the -P specifies to report only on CPU core 1 (a quad-core would have 0-3); use **-P ALL** for all cores
**$ sar -r -f /var/log/sa/sa10** - most queries use the same syntax as above - this one querying memory stats with **-r**
**$ sar -p -d** - This query for block devices uses the -p option ("pretty print") so drives listed have familiar names to us.

General options are, **-f** to specify a file (day), **ALL** for extended info or all items (in context), **-p** for human readable, then numbers to specify frequency to run followed by how many times to run before exiting. There is also an option to put in a start time, like **sar -s 09:30:00** for 9:30AM, can be used with **-f** to specify a day. Optional end time with **-e**

The most often used queries are **-u** (CPU), **-r** (memory), **-S** (swap), **-b** (I/O stats), **-d** (block devices), **-n** (network)
*Here are some others:*
- **-q** - run queue size and load average
- **-w** - processes created per second, and total number of context switches per second
- **-R** - number of memory pages freed, used, and cached per second by the system.
- **-B** - paging statistics. i.e Number of KB paged in (and out) from disk per second.
- **-W** - page swap statistics. i.e Page swap in (and out) per second.

Network activity reporting has a ton of options; **sar -n <keyword>** - where <keyword> can be any of these:

| | |
|---|---|
| DEV - network devices vital stats for eth0, eth1 | ICMP - ICMPv4 network traffic |
| EDEV - network device failure stats | EICMP - ICMPv4 network errors |
| NFS - NFS client activities | TCP - TCPv4 network traffic |
| NFSD - NFS server activities | ETCP - TCPv4 network errors |
| SOCK - sockets in use for IPv4 | UDP - UDPv4 network traffic |
| IP - IPv4 network traffic | SOCK6, IP6, EIP6, ICMP6, UDP6 for IPv6 |
| EIP - IPv4 network errors | ALL - all options in one output- very long. |

## *vmstat - Memory, Processes, Paging etc*

**vmstat -a** option also displays active and inactive memory. Display units in KB and MB with **--unit k** or **m**
**vmstat 2 5** - at every 2 second interval, we want to see 5 polling groups (rows of data). Just like iostat, the first line is going to be bigger because it is generic overview of the last period of total system activity

| proc | | | memory | | | | swap | | io | | system | | cpu | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| r | b | w | swpd | free | buff | cache | si | so | bi | bo | in | cs | us | sy | id | wa |
| 2 | 0 | 0 | 120 | 30180 | 1736 | 46696 | 0 | 0 | 424 | 25 | 134 | 77 | 5 | 9 | 86 | |

*Field definitions:*
Proc - r: Running waiting for CPU; b: Uninterruptable sleeping processes (waiting for I/O); w: Swapped out processes
Memory - swpd: swap areas (in /proc/swaps); free: Idle memory; buff: buffers; cache: cache used.
Swap - si/so: Memory swapped in and out from disk - use to get more info when top shows huge swap usage
IO - bi/bo: Blocks received from/ to block device (like a hard disk)
System - in: interrupts/ IRQs per second, including the clock; cs: context switches per second. (when the CPU switches tasks among programs)
CPU - us: user time, non-kernel code including nice time; sy: kernel code; system time - network, IO interrupts, etc; id: idle time. Prior to Linux 2.5.41, including IO-wait time.

**vmstat -s** helps too- lots more info dumped like this:

```
[root@server1 ~]# vmstat -s
      1878184 K total memory
      1804284 K used memory
       329884 K active memory
      1273128 K inactive memory
        73900 K free memory
      1021104 K buffer memory
       151208 K swap cache
      2097148 K total swap
            0 K used swap
      2097148 K free swap
        12824 non-nice user cpu ticks
            3 nice user cpu ticks
         7702 system cpu ticks
       410614 idle cpu ticks
          405 IO-wait cpu ticks
            1 IRQ cpu ticks
         1113 softirq cpu ticks
            0 stolen cpu ticks
     28249840 pages paged in
        49102 pages paged out
            0 pages swapped in
            0 pages swapped out
       552140 interrupts
      1095772 CPU context switches
   1409692776 boot time
         5257 forks
```

Resident memory is the physical memory. swap in/ out- if you see in **top** swap is being used you can check here to see if swap is being used actively. Similarly, bi/bo is blocked in/out so you can tell if blocked processes are spending time reading more or writing. Info is grabbed from **/proc/meminfo**, **/proc/stat**, **/proc/*/stat**

**Regular mode is simply "VM mode", but there are others:**
*Disk Mode* [ -d, --disk ] Use [ -D, --disk-sum ] for summary statistics
displays these fields for both reads and writes: total (total completed successfully), merged (grouped reads or writes, resulting in one I/O), sectors (sectors read or written successfully), ms (milliseconds spent reading, or writing). Then also displays cur (for I/O currently in progress, and s (seconds spent for I/O)

*Disk Partition Mode* [ -p, --partition device ] - Detailed statistics about partition (kernel 2.5.70 or above required). Fields include: reads (reads to this partition), read sectors (read sectors for partition), writes (total number of writes issued), requested writes (number of write requests made)

*Slab Mode* [ -m, --slabs ] - Displays the contents of **/proc/slabinfo** Fields include cache (cache name), num (number of currently active objects), total (number of available objects), size (of each object) and pages (the number of pages with at least one active object)

## *iostat - CPU and I/O Stats for Devices and Partitions*

**iostat [option] [interval] [count]**
Interval specifies time in seconds between each report, and count specifies the number of reports generated before exiting. **iostat 2 5 - indicates** a 2 sec interval, 5 iterations. If you don't specify iterations it will run continuous.
*When run for the first time, the first report contains information since the system was boot, while each subsequent report covers the time period since the last report was generated, so you may want to ignore the first one.*

The first part of the normal output reports the CPU utilization, and should be self-explanatory from other tools like **top**
The last two columns show %CPU time idle with an outstanding disk I/O request (lag waiting for disk activity) and without an outstanding disk I/O request. If you only want to get this top part of the report, run **iostat -c**

Linux 2.6.31-17-generic (drt-laptop) 03/24/16 _ii686_ (1 CPU)

| avg-cpu: | %user | %nice | %system | %iowait | %idle |
|---|---|---|---|---|---|
| | 25.99 | 0.78 | 7.43 | 12.77 | 53.03 |

| Device: | tps | Blk_read/s | Blk_wrtn/s | Blk_read | Blk_wrtn |
|---|---|---|---|---|---|
| sda | 27.40 | 797.19 | 201.27 | 800902 | 202208 |
| sr0 | 0.03 | 1.24 | 0.00 | 1248 | 0 |

The device report has transfers per second, number of blocks per second read, number of blocks per second written, total number of blocks read, and total number of blocks written. You can use the **-k** (kilobytes) or **-m** (megabytes) parameter to change the last four columns to be expressed in human-readable terms instead of blocks
For only the device report above, run with **iostat -d**
**iostat -x** gives stats in an extended report.
Adding a device identifier with **/dev/sdaX** syntax will limit the report to the specified device
Using **-p** in there will show data on partitions, and **-N** gives LVM names and stats.
For NFS info, use the **iostat -n** option. **iostat -z** omits inactive devices.
Like anything, do output redirection into a file, and/or use awk to extract columns you need.
Uses **/proc/stat**, **/proc/uptime**, **/proc/partitions**, **/proc/diskstats**, **/sys**, **/proc/self/mountstats**

## *mpstat - Multiprocessor Stats*

If you don't specify a CPU, it will spit out a global average of all of them. Specify a processor with **-P** and put ALL to get stats on all CPUs.
**$ mpstat -P ALL**

| 1:30:26 | IST | CPU | %usr | %nice | %sys | %iowait | %irq | %soft | %steal | %guest | %gnice | %idle |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1:30:26 | IST | all | 37.33 | 0.01 | 4.57 | 2.58 | 0.00 | 0.07 | 0.00 | 0.00 | 0.00 | 55.44 |
| 1:30:26 | IST | 0 | 37.90 | 0.01 | 4.96 | 2.62 | 0.00 | 0.03 | 0.00 | 0.00 | 0.00 | 54.48 |

**$ mpstat -P ALL 2 5** - Just like with iostat - add iteration and interval count The option **-I** shows interrupt stats for each processor, and **-u** says CPU stats in case the other options are used.

*Easiest way: **$ mpstat -A** is the quick and easy equivalent of mpstat **-I ALL -u -P ALL***

## *pidstat - Stats on Processes*

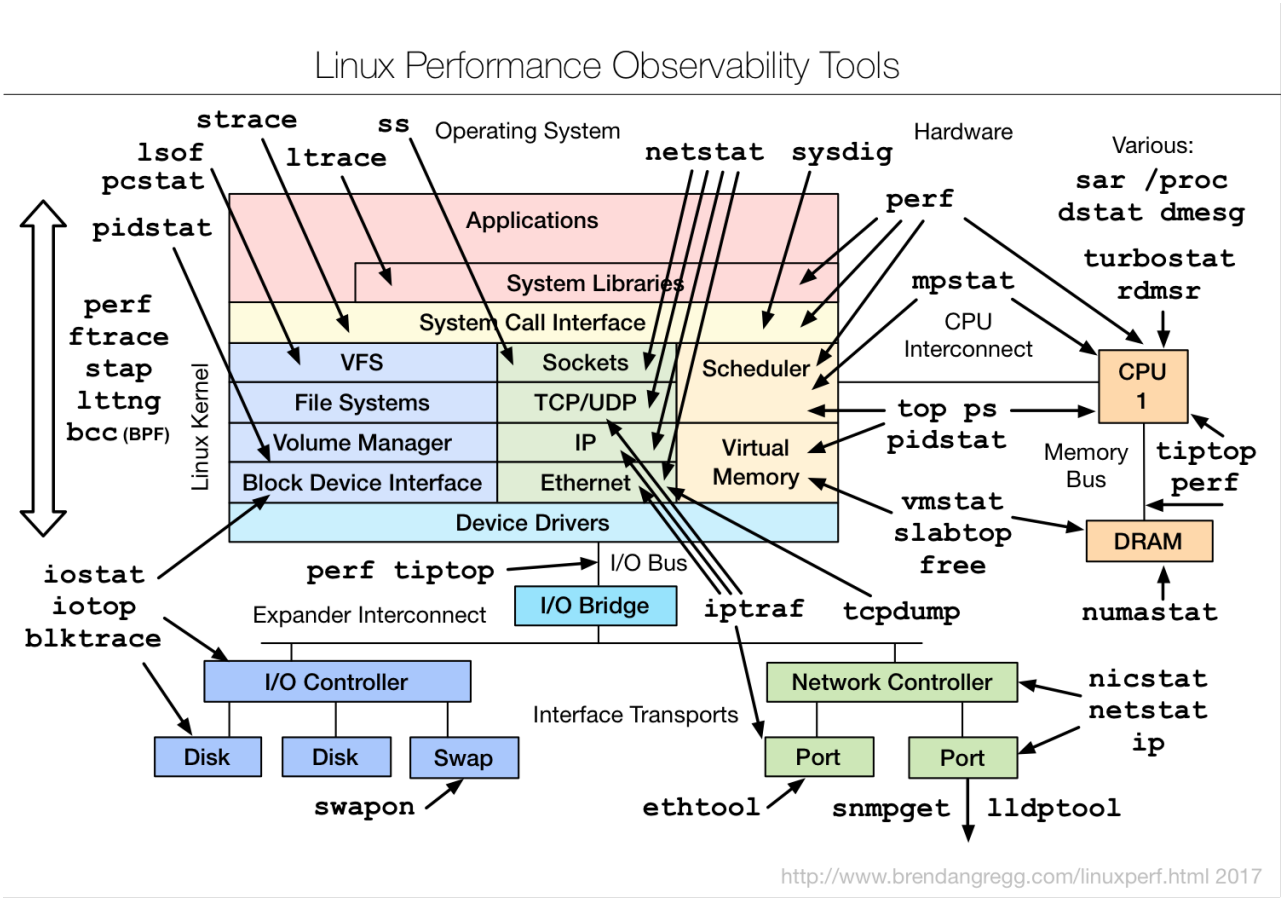
**pidstat 2 5** - five reports of CPU statistics for every active task in the system at two second intervals.
**pidstat -r** - for a focus on page faults in memory statistics:

| # Time | UID | PID | minflt/s | majflt/s | VSZ | RSS | %MEM | Command |
|---|---|---|---|---|---|---|---|---|
| 1409816695 | 1000 | 3958 | 3378.22 | 0.00 | 707420 | 215972 | 5.32 | cinnamon |

**pidstat -d** option for disk I/O stats:

| | PID | kB_rd/s | kB_wr/s | kB_ccwr/s | Command |
|---|---|---|---|---|---|
| 03:27:03 EDT | 1 | 0.00 | 8.00 | 2.00 | init |

**pidstat -p 1643** for PID 1643. Use **-p ALL** for all processes
Enter a string to match a process name like **-C httpd**; start an executable with **pidstat -e <command>** to monitor it.
**pidstat -T CHILD -p 1643** - for the child processes of PID 1643. Only child processes with non-zero statistics values are displayed. **-T ALL** means all tasks and child processes.
**pidstat** with no options assumes **-p ALL**
For realtime stats use the **-R** option, or **-h** to output one horizontal line of output to easily parse with **awk**, etc.




***Strongly recommended: Brendan Gregg's work: http://www.brendangregg.com/linuxperf.html***