

## 1. SAR

Using sar utility you can do two things: 1) Monitor system real time performance (CPU, Memory, I/O, etc) 2) Collect performance data in the background on an on-going basis and do analysis on the historical data to identify bottlenecks.

Sar is part of the sysstat package. The following are some of the things you can do using sar utility.

- Collective CPU usage
- Individual CPU statistics
- Memory used and available
- Swap space used and available
- Overall I/O activities of the system
- Individual device I/O activities
- Context switch statistics
- Run queue and load average data
- Network statistics
- Report sar data from a specific time
- and lot more..

The following sar command will display the system CPU statistics 3 times (with 1 second interval).

The following “sar -b” command reports I/O statistics. “1 3” indicates that the sar -b will be executed for every 1 second for a total of 3 times.

```
$ sar -b 1 3
```

```
Linux 2.6.18-194.el5PAE (dev-db)      03/26/2011      _i686_ (8 CPU)
```

01:56:28 PM	tps	rtps	wtps	bread/s	bwrtn/s
01:56:29 PM	346.00	264.00	82.00	2208.00	768.00
01:56:30 PM	100.00	36.00	64.00	304.00	816.00
01:56:31 PM	282.83	32.32	250.51	258.59	2537.37
Average:	242.81	111.04	131.77	925.75	1369.90

More SAR examples: [How to Install/Configure Sar \(sysstat\) and 10 Useful Sar Command Examples](#)

## 2. Tcpdump

tcpdump is a network packet analyzer. Using tcpdump you can capture the packets and analyze it for any performance bottlenecks.

The following tcpdump command example displays captured packets in ASCII.

```
$ tcpdump -A -i eth0
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
```

```
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

```
14:34:50.913995 IP valh4.lell.net.ssh > yy.domain.innetbcp.net.11006: P  
1457239478:1457239594(116) ack 1561461262 win 63652
```

```
E.....@.@..].i...9...*.V...].P....h....E...>{..U=...g.
```

```
.....G..7\+KA....A...L.
```

```
14:34:51.423640 IP valh4.lell.net.ssh > yy.domain.innetbcp.net.11006: P  
116:232(116) ack 1 win 63652
```

```
E.....@.@..\...i...9...*.V...*].P....h....7.....X...!....Im.S.g.u:*..O&....^#Ba...
```

```
E..(R.@.|.....9...i.*...].V...*P..OWp.....
```

Using tcpdump you can capture packets based on several custom conditions. For example, capture packets that flow through a particular port, capture tcp communication between two specific hosts, capture packets that belongs to a specific protocol type, etc.

More tcpdump examples: [15 TCPDUMP Command Examples](#)

### 3. Nagios

Nagios is an open source monitoring solution that can monitor pretty much anything in your IT infrastructure. For example, when a server goes down it can send a notification to your sysadmin team, when a database goes down it can page your DBA team, when the a web server goes down it can notify the appropriate team.

You can also set warning and critical threshold level for various services to help you proactively address the issue. For example, it can notify sysadmin team when a disk partition becomes 80% full, which will give enough time for the sysadmin team to work on adding more space before the issue becomes critical.

Nagios also has a very good user interface from where you can monitor the health of your overall IT infrastructure.

The following are some of the things you can monitor using Nagios:

- Any hardware (servers, switches, routers, etc)
- Linux servers and Windows servers
- Databases (Oracle, MySQL, PostgreSQL, etc)
- Various services running on your OS (sendmail, nis, nfs, ldap, etc)
- Web servers

- Your custom application
- etc.

More Nagios examples: [How to install and configure Nagios](#), [monitor remote Windows machine](#), and [monitor remote Linux server](#).

#### 4. Iostat

iostat reports CPU, disk I/O, and NFS statistics. The following are some of iostat command examples.

Iostat without any argument displays information about the CPU usage, and I/O statistics about all the partitions on the system as shown below.

```
$ iostat

Linux 2.6.32-100.28.5.el6.x86_64 (dev-db)      07/09/2011

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           5.68    0.00    0.52    2.03    0.00   91.76

Device:            tps    Blk_read/s    Blk_wrtn/s    Blk_read    Blk_wrtn
sda                 194.72      1096.66      1598.70  2719068704  3963827344
sda1                178.20       773.45      1329.09  1917686794  3295354888
sda2                 16.51       323.19       269.61   801326686   668472456
sdb                  371.31       945.97      1073.33  2345452365  2661206408
sdb1                 371.31       945.95      1073.33  2345396901  2661206408
sdc                  408.03       207.05       972.42   513364213  2411023092
sdc1                 408.03       207.03       972.42   513308749  2411023092
```

By default iostat displays I/O data for all the disks available in the system. To view statistics for a specific device (For example, /dev/sda), use the option -p as shown below.

```
$ iostat -p sda
```

Linux 2.6.32-100.28.5.el6.x86\_64 (dev-db) 07/09/2011

avg-cpu:	%user	%nice	%system	%iowait	%steal	%idle
	5.68	0.00	0.52	2.03	0.00	91.76

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	194.69	1096.51	1598.48	2719069928	3963829584
sda2	336.38	27.17	54.00	67365064	133905080
sda1	821.89	0.69	243.53	1720833	603892838

## 5. Mpstat

mpstat reports processors statistics. The following are some of mpstat command examples.

Option -A, displays all the information that can be displayed by the mpstat command as shown below. This is really equivalent to “mpstat -I ALL -u -P ALL” command.

```
$ mpstat -A
```

Linux 2.6.32-100.28.5.el6.x86\_64 (dev-db) 07/09/2011 \_x86\_64\_ (4 CPU)

10:26:34 PM	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%idle
-------------	-----	------	-------	------	---------	------	-------	--------	--------	-------

10:26:34 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	99.99
-------------	-----	------	------	------	------	------	------	------	------	-------

10:26:34 PM	0	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	99.98
-------------	---	------	------	------	------	------	------	------	------	-------

10:26:34 PM	1	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	99.98
-------------	---	------	------	------	------	------	------	------	------	-------

10:26:34 PM	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100.00										

10:26:34 PM	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100.00										

10:26:34 PM	CPU	intr/s
-------------	-----	--------

10:26:34 PM	all	36.51
-------------	-----	-------

10:26:34 PM	0	0.00
-------------	---	------

10:26:34 PM	1	0.00
-------------	---	------

10:26:34 PM	2	0.04
-------------	---	------

10:26:34 PM	3	0.00
-------------	---	------

10:26:34 PM	CPU	0/s	1/s	8/s	9/s	12/s	14/s	15/s	16/s
19/s	20/s	21/s	33/s	NMI/s	LOC/s	SPU/s	PMI/s	PND/s	RES/s
CAL/s	TLB/s	TRM/s	THR/s	MCE/s	MCP/s	ERR/s	MIS/s		

10:26:34 PM	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	7.47	0.00	0.00	0.00	0.00
0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00		

10:26:34 PM	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	4.90	0.00	0.00	0.00	0.00
0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00		

10:26:34 PM	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
0.00	0.00	0.00	0.00	0.00	3.32	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		

10:26:34 PM	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.									

mpstat Option -P ALL, displays all the individual CPUs (or Cores) along with its statistics as shown below.

```
$ mpstat -P ALL
```

```
Linux 2.6.32-100.28.5.el6.x86_64 (dev-db)      07/09/2011      _x86_64_      (4 CPU)
```

```
10:28:04 PM  CPU      %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest
%idle
```

```
10:28:04 PM  all      0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
99.99
```

```
10:28:04 PM    0      0.01    0.00    0.01    0.01    0.00    0.00    0.00    0.00
99.98
```

```
10:28:04 PM    1      0.00    0.00    0.01    0.00    0.00    0.00    0.00    0.00
99.98
```

```
10:28:04 PM    2      0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
100.00
```

```
10:28:04 PM    3      0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
100.00
```

## 6. Vmstat

vmstat reports virtual memory statistics. The following are some of vmstat command examples.

vmstat by default will display the memory usage (including swap) as shown below.

```
$ vmstat
```

```
procs -----memory----- ---swap-- -----io---- --system-- -----cpu-----
```

```
 r  b   swpd   free   buff  cache   si   so    bi    bo    in   cs us sy id wa st
```

```
 0   0 305416 260688  29160 2356920    2    2     4     1    0    0  6  1 92  2  0
```

To execute vmstat every 2 seconds for 10 times, do the following. After executing 10 times, it will stop automatically.

```
$ vmstat 2 10
```

```
procs -----memory----- ---swap-- -----io---- --system-- -----cpu-----
```

r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
1	0	0	537144	182736	6789320	0	0	0	0	1	1	0	0	100	0	0
0	0	0	537004	182736	6789320	0	0	0	0	50	32	0	0	100	0	0
..																

iostat and vmstat are part of the sar utility. You should install sysstat package to get iostat and vmstat working.

More examples: [24 iostat, vmstat and mpstat command Examples](#)

## 7. PS Command

Process is a running instance of a program. Linux is a multitasking operating system, which means that more than one process can be active at once. Use ps command to find out what processes are running on your system.

ps command also give you lot of additional information about the running process which will help you identify any performance bottlenecks on your system.

The following are few ps command examples.

Use -u option to display the process that belongs to a specific username. When you have multiple username, separate them using a comma. The example below displays all the process that are owned by user wwwrun, or postfix.

```
$ ps -f -u wwwrun,postfix
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
postfix	7457	7435	0	Mar09	?	00:00:00	qmgr -l -t fifo -u
wwwrun	7495	7491	0	Mar09	?	00:00:00	/usr/sbin/httpd2-prefork -f /etc/apache2/httpd.conf
wwwrun	7496	7491	0	Mar09	?	00:00:00	/usr/sbin/httpd2-prefork -f /etc/apache2/httpd.conf
wwwrun	7497	7491	0	Mar09	?	00:00:00	/usr/sbin/httpd2-prefork -f /etc/apache2/httpd.conf
wwwrun	7498	7491	0	Mar09	?	00:00:00	/usr/sbin/httpd2-prefork -f /etc/apache2/httpd.conf

```

wwwrun    7499  7491  0 Mar09 ?           00:00:00 /usr/sbin/httpd2-prefork -f
/etc/apache2/httpd.conf

wwwrun   10078  7491  0 Mar09 ?           00:00:00 /usr/sbin/httpd2-prefork -f
/etc/apache2/httpd.conf

wwwrun   10082  7491  0 Mar09 ?           00:00:00 /usr/sbin/httpd2-prefork -f
/etc/apache2/httpd.conf

postfix  15677  7435  0 22:23 ?           00:00:00 pickup -l -t fifo -u

```

The example below display the process Id and commands in a hierarchy. `--forest` is an argument to `ps` command which displays ASCII art of process tree. From this tree, we can identify which is the parent process and the child processes it forked in a recursive manner.

```

$ ps -e -o pid,args --forest

 468  \_ sshd: root@pts/7

 514  |   \_ -bash

17484  \_ sshd: root@pts/11

17513  |   \_ -bash

24004  |       \_ vi ./790310__11117/journal

15513  \_ sshd: root@pts/1

15522  |   \_ -bash

 4280  \_ sshd: root@pts/5

 4302  |   \_ -bash

```

More ps examples: [7 Practical PS Command Examples for Process Monitoring](#)

## 8. Free

`Free` command displays information about the physical (RAM) and swap memory of your system.

In the example below, the total physical memory on this system is 1GB. The values displayed below are in KB.



```
# free

            total    used    free   shared  buffers   cached

Mem: 1034624    1006696  27928    0         174136   615892

-/+ buffers/cache:    216668    817956

Swap:    2031608         0    2031608
```

The following example will display the total memory on your system including RAM and Swap.

In the following command:

- option m displays the values in MB
- option t displays the “Total” line, which is sum of physical and swap memory values
- option o is to hide the buffers/cache line from the above example.

```
# free -mto

            total        used        free        shared        buffers        cached

Mem:         1010         983           27             0           170           601

Swap:         1983             0        1983

Total:         2994         983        2011
```

## 9. TOP

Top command displays all the running process in the system ordered by certain columns. This displays the information real-time.

You can kill a process without exiting from top. Once you’ve located a process that needs to be killed, press “k” which will ask for the process id, and signal to send. If you have the privilege to kill that particular PID, it will get killed successfully.

```
PID to kill: 1309
```

```
Kill PID 1309 with signal [15]:
```

```

PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
1309 geek      23   0 2483m 1.7g  27m S    0 21.8  45:31.32 gagent
```

```
1882 geek    25    0 2485m 1.7g  26m S    0 21.7  22:38.97 gagent
5136 root     16    0 38040  14m 9836 S    0  0.2   0:00.39 nautilus
```

Use top -u to display a specific user processes only in the top command output.

```
$ top -u geek
```

While unix top command is running, press u which will ask for username as shown below.

```
Which user (blank for all): geek
```

```

PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
1309 geek      23   0 2483m 1.7g  27m S   0 21.8  45:31.32 gagent
1882 geek      25   0 2485m 1.7g  26m S   0 21.7  22:38.97 gagent
```

More top examples: [15 Practical Linux Top Command Examples](#)

## 10. Pmap

pmap command displays the memory map of a given process. You need to pass the pid as an argument to the pmap command.

The following example displays the memory map of the current bash shell. In this example, 5732 is the PID of the bash shell.

```
$ pmap 5732

5732:  -bash

00393000    104K r-x--  /lib/ld-2.5.so
003b1000   1272K r-x--  /lib/libc-2.5.so
00520000      8K r-x--  /lib/libdl-2.5.so
0053f000    12K r-x--  /lib/libtermcap.so.2.0.8
0084d000    76K r-x--  /lib/libnsl-2.5.so
```

```

00c57000    32K r-x--  /lib/libnss\_nis-2.5.so

00c8d000    36K r-x--  /lib/libnss\_files-2.5.so

b7d6c000   2048K r----- /usr/lib/locale/locale-archive

bfd10000     84K rw---   [ stack ]

total      4796K

```

`pmap -x` gives some additional information about the memory maps.

```

$ pmap -x 5732

5732:  -bash

Address      Kbytes      RSS      Anon  Locked  Mode    Mapping
-----
00393000      104        -        -      -    r-x--  ld-2.5.so
003b1000     1272        -        -      -    r-x--  libc-2.5.so
00520000         8        -        -      -    r-x--  libdl-2.5.so
0053f000        12        -        -      -    r-x--  libtermcap.so.2.0.8
0084d000        76        -        -      -    r-x--  libnsl-2.5.so
00c57000        32        -        -      -    r-x--  libnss\_nis-2.5.so
00c8d000        36        -        -      -    r-x--  libnss\_files-2.5.so
b7d6c000     2048        -        -      -    r----- locale-archive
bfd10000        84        -        -      -    rw---   [ stack ]

-----

total kB      4796        -        -      -

```

To display the device information of the process maps use ‘`pmap -d pid`’.

## 11. Netstat

Netstat command displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc.,

The following are some netstat command examples.

List all ports (both listening and non listening) using netstat -a as shown below.

```
# netstat -a | more
```

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	localhost:30037	*:*	LISTEN
udp	0	0	*:bootpc	*:*	

Active UNIX domain sockets (servers and established)

Proto	RefCnt	Flags	Type	State	I-Node	Path
unix	2	[ ACC ]	STREAM	LISTENING	6135	/tmp/.X11-unix/X0
unix	2	[ ACC ]	STREAM	LISTENING	5140	/var/run/acpid.socket

Use the following netstat command to find out on which port a program is running.

```
# netstat -ap | grep ssh
```

(Not all processes could be identified, non-owned process info

will not be shown, you would have to be root to see it all.)

tcp	1	0	dev-db:ssh	<a href="#">101.174.100.22:39213</a>	CLOSE_WAIT	-
tcp	1	0	dev-db:ssh	<a href="#">101.174.100.22:57643</a>	CLOSE_WAIT	-

Use the following netstat command to find out which process is using a particular port.

```
# netstat -an | grep ':80'
```

More netstat examples: [10 Netstat Command Examples](#)

## 12. IPTraf

IPTraf is a IP Network Monitoring Software. The following are some of the main features of IPTraf:

- It is a console based (text-based) utility.
- This displays IP traffic crossing over your network. This displays TCP flag, packet and byte counts, ICMP, OSPF packet types, etc.
- Displays extended interface statistics (including IP, TCP, UDP, ICMP, packet size and count, checksum errors, etc.)
- LAN module discovers hosts automatically and displays their activities
- Protocol display filters to view selective protocol traffic
- Advanced Logging features
- Apart from ethernet interface it also supports FDDI, ISDN, SLIP, PPP, and loopback
- You can also run the utility in full screen mode. This also has a text-based menu.

More info: [IPTraf Home Page](#). [IPTraf screenshot](#).

## 13. Strace

Strace is used for debugging and troubleshooting the execution of an executable on Linux environment. It displays the system calls used by the process, and the signals received by the process.

Strace monitors the system calls and signals of a specific program. It is helpful when you do not have the source code and would like to debug the execution of a program. strace provides you the execution sequence of a binary from start to end.

### Trace a Specific System Calls in an Executable Using Option -e

By default, strace displays all system calls for the given executable. The following example shows the output of strace for the Linux ls command.

```
$ strace ls

execve("/bin/ls", ["ls"], [/* 21 vars */]) = 0

brk(0)                                = 0x8c31000

access("/etc/ld.so.nohwcap", F_OK)     = -1 ENOENT (No such file or directory)

mmap2(NULL, 8192, PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb78c7000

access("/etc/ld.so.preload", R_OK)     = -1 ENOENT (No such file or directory)

open("/etc/ld.so.cache", O_RDONLY)     = 3
```

```
fstat64(3, {st_mode=S_IFREG|0644, st_size=65354, ...}) = 0
```

To display only a specific system call, use the strace -e option as shown below.

```
$ strace -e open ls

open("/etc/ld.so.cache", O_RDONLY)      = 3

open("/lib/libselinux.so.1", O_RDONLY) = 3

open("/lib/librt.so.1", O_RDONLY)       = 3

open("/lib/libacl.so.1", O_RDONLY)      = 3

open("/lib/libc.so.6", O_RDONLY)        = 3

open("/lib/libdl.so.2", O_RDONLY)       = 3

open("/lib/libpthread.so.0", O_RDONLY)  = 3

open("/lib/libattr.so.1", O_RDONLY)     = 3

open("/proc/filesystems", O_RDONLY|O_LARGEFILE) = 3

open("/usr/lib/locale/locale-archive", O_RDONLY|O_LARGEFILE) = 3

open(".", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_CLOEXEC) = 3
```

More strace examples: [7 Strace Examples to Debug the Execution of a Program in Linux](#)

#### 14. Lsof

Lsof stands for ls open files, which will list all the open files in the system. The open files include network connection, devices and directories. The output of the lsof command will have the following columns:

- COMMAND process name.
- PID process ID
- USER Username
- FD file descriptor
- TYPE node type of the file
- DEVICE device number
- SIZE file size
- NODE node number
- NAME full path of the file name.

To view all open files of the system, execute the lsof command without any parameter as shown below.

```
# lsof | more
```

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
init	1	root	cwd	DIR	8,1	4096	2	/
init	1	root	rtd	DIR	8,1	4096	2	/
init /sbin/init	1	root	txt	REG	8,1	32684	983101	
init	1	root	mem	REG	8,1	106397	166798	/lib/ <a href="#">ld-2.3.4.so</a>
init /lib/tls/ <a href="#">libc-2.3.4.so</a>	1	root	mem	REG	8,1	1454802	166799	
init /lib/libsepol.so.1	1	root	mem	REG	8,1	53736	163964	
init /lib/libselinux.so.1	1	root	mem	REG	8,1	56328	166811	
init /dev/initctl	1	root	10u	FIFO	0,13		972	
migration	2	root	cwd	DIR	8,1	4096	2	/

skipped..

To view open files by a specific user, use lsof -u option to display all the files opened by a specific user.

```
# lsof -u ramesh
```

vi	7190	ramesh	txt	REG	8,1	474608	475196	/bin/vi
sshd	7163	ramesh	3u	IPv6	15088263			TCP dev-db:ssh->abc-12-12-12-12.

To list users of a particular file, use lsof as shown below. In this example, it displays all users who are currently using vi.

```
# lsof /bin/vi
```

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
vi	7258	root	txt	REG	8,1	474608	475196	/bin/vi
vi	7300	ramesh	txt	REG	8,1	474608	475196	/bin/vi

## 15. Ntop

Ntop is just like top, but for network traffic. ntop is a network traffic monitor that displays the network usage.

You can also access ntop from browser to get the traffic information and network status.

The following are some the key features of ntop:

- Display network traffic broken down by protocols
- Sort the network traffic output based on several criteria
- Display network traffic statistics
- Ability to store the network traffic statistics using RRD
- Identify the identify of the users, and host os
- Ability to analyze and display IT traffic
- Ability to work as NetFlow/sFlow collector for routers and switches
- Displays network traffic statistics similar to RMON
- Works on Linux, MacOS and Windows

More info: [Ntop home page](#)

## 16. GkrellM

GKrellM stands for GNU Krell Monitors, or GTK Krell Meters. It is GTK+ toolkit based monitoring program, that monitors various sytem resources. The UI is stakable. i.e you can add as many monitoring objects you want one on top of another. Just like any other desktop UI based monitoring tools, it can monitor CPU, memory, file system, network usage, etc. But using plugins you can monitoring external applications.

More info: [GkrellM home page](#)

## 17. w and uptime

While monitoring system performance, w command will hlep to know who is logged on to the system.

```
$ w
```

```
09:35:06 up 21 days, 23:28,  2 users,  load average: 0.00, 0.00, 0.00
```

USER	TTY	FROM	LOGIN@	IDLE	JCPU	PCPU	WHAT
------	-----	------	--------	------	------	------	------



```

root      tty1      :0              24Oct11  21days 1:05   1:05 /usr/bin/Xorg :0 -nr -
verbose

ramesh    pts/0      192.168.1.10   Mon14    0.00s   15.55s 0.26s sshd: localuser
[priv]

john      pts/0      192.168.1.11   Mon07    0.00s   19.05s 0.20s sshd: localuser
[priv]

jason     pts/0      192.168.1.12   Mon07    0.00s   21.15s 0.16s sshd: localuser
[priv]

```

For each and every user who is logged on, it displays the following info:

- Username
- tty info
- Remote host ip-address
- Login time of the user
- How long the user has been idle
- JCPU and PCUP
- The command of the current process the user is executing

Line 1 of the w command output is similar to the uptime command output. It displays the following:

- Current time
- How long the system has been up and running
- Total number of users who are currently logged on the system
- Load average for the last 1, 5 and 15 minutes

If you want only the uptime information, use the uptime command.

```

$ uptime

09:35:02 up 106 days, 28 min,  2 users,  load average: 0.08, 0.11, 0.05

```

Please note that both w and uptime command gets the information from the /var/run/utmp data file.

## 18. /proc

/proc is a virtual file system. For example, if you do `ls -l /proc/stat`, you'll notice that it has a size of 0 bytes, but if you do `cat /proc/stat`, you'll see some content inside the file.

Do a `ls -l /proc`, and you'll see lot of directories with just numbers. These numbers represents the process ids, the files inside this numbered directory corresponds to the process with that particular PID.

The following are the important files located under each numbered directory (for each process):

- cmdline – command line of the command.

- environ – environment variables.
- fd – Contains the file descriptors which is linked to the appropriate files.
- limits – Contains the information about the specific limits to the process.
- mounts – mount related information

The following are the important links under each numbered directory (for each process):

- cwd – Link to current working directory of the process.
- exe – Link to executable of the process.
- root – Link to the root directory of the process.

More /proc examples: [Explore Linux /proc File System](#)

## 19. KDE System Guard

This is also called as KSysGuard. On Linux desktops that run KDE, you can use this tool to monitor system resources. Apart from monitoring the local system, this can also monitor remote systems.

If you are running KDE desktop, go to Applications -> System -> System Monitor, which will launch the KSysGuard. You can also type ksysguard from the command line to launch it.

This tool displays the following two tabs:

- Process Table – Displays all active processes. You can sort, kill, or change priority of the processes from here
- System Load – Displays graphs for CPU, Memory, and Network usages. These graphs can be customized by right clicking on any of these graphs.

To connect to a remote host and monitor it, click on File menu -> Monitor Remote Machine -> specify the ip-address of the host, the connection method (for example, ssh). This will ask you for the username/password on the remote machine. Once connected, this will display the system usage of the remote machine in the Process Table and System Load tabs.

## 20. GNOME System Monitor

On Linux desktops that run GNOME, you can use the this tool to monitor processes, system resources, and file systems from a graphical interface. Apart from monitoring, you can also use this UI tool to kill a process, change the priority of a process.

If you are running GNOME desktop, go to System -> Administration -> System Monitor, which will launch the GNOME System Monitor. You can also type gnome-system-monitor from the command line to launch it.

This tool has the following four tabs:

- System – Displays the system information including Linux distribution version, system resources, and hardware information.
- Processes – Displays all active processes that can be sorted based on various fields
- Resources – Displays CPU, memory and network usages
- File Systems – Displays information about currently mounted file systems

More info: [GNOME System Monitor home page](#)

## 21. Conky

Conky is a system monitor or X. Conky displays information in the UI using what it calls objects. By default there are more than 250 objects that are bundled with conky, which displays various

monitoring information (CPU, memory, network, disk, etc.). It supports IMAP, POP3, several audio players.

You can monitor and display any external application by creating your own objects using scripting. The monitoring information can be displayed in various formats: Text, graphs, progress bars, etc. This utility is extremely configurable.

More info: [Conky screenshots](#)

## 22. Cacti

Cacti is a PHP based UI frontend for the RRDTool. Cacti stores the data required to generate the graph in a MySQL database.

The following are some high-level features of Cacti:

- Ability to perform the data gathering and store it in MySQL database (or round robin archives)
- Several advanced graphing features are available (grouping of GPRINT graph items, auto-padding for graphs, manipulate graph data using CDEF math function, all RRDTool graph items are supported)
- The data source can gather local or remote data for the graph
- Ability to fully customize Round robin archive (RRA) settings
- User can define custom scripts to gather data
- SNMP support (php-snmp, ucd-snmp, or net-snmp) for data gathering
- Built-in poller helps to execute custom scripts, get SNMP data, update RRD files, etc.
- Highly flexible graph template features
- User friendly and customizable graph display options
- Create different users with various permission sets to access the cacti frontend
- Granular permission levels can be set for the individual user
- and lot more..

More info: [Cacti home page](#)

## 23. Vnstat

vnstat is a command line utility that displays and logs network traffic of the interfaces on your systems. This depends on the network statistics provided by the kernel. So, vnstat doesn't add any additional load to your system for monitoring and logging the network traffic.

vnstat without any argument will give you a quick summary with the following info:

- The last time when the vnStat database located under /var/lib/vnstat/ was updated
- From when it started collecting the statistics for a specific interface
- The network statistic data (bytes transmitted, bytes received) for the last two months, and last two days.

```
# vnstat
```

```
Database updated: Sat Oct 15 11:54:00 2011
```

```
eth0 since 10/01/11
```

```
rx: 12.89 MiB    tx: 6.94 MiB    total: 19.82 MiB
```

monthly

	rx		tx		total		avg. rate
-----+-----+-----+-----							
Sep '11	12.90 MiB		6.90 MiB		19.81 MiB		0.14 kbit/s
Oct '11	12.89 MiB		6.94 MiB		19.82 MiB		0.15 kbit/s
-----+-----+-----+-----							
estimated	29 MiB		14 MiB		43 MiB		

daily

	rx		tx		total		avg. rate
-----+-----+-----+-----							
yesterday	4.30 MiB		2.42 MiB		6.72 MiB		0.64 kbit/s
today	2.03 MiB		1.07 MiB		3.10 MiB		0.59 kbit/s
-----+-----+-----+-----							
estimated	4 MiB		2 MiB		6 MiB		

Use “vnstat -t” or “vnstat -top10” to display all time top 10 traffic days.

```
$ vnstat --top10
```

```
eth0 / top 10
```

#	day	rx		tx		total		avg. rate
-----+-----+-----+-----								
1	10/12/11	4.30 MiB		2.42 MiB		6.72 MiB		0.64 kbit/s
2	10/11/11	4.07 MiB		2.17 MiB		6.24 MiB		0.59 kbit/s
3	10/10/11	2.48 MiB		1.28 MiB		3.76 MiB		0.36 kbit/s
....								
-----+-----+-----+-----								

More vnstat Examples: [How to Monitor and Log Network Traffic using VNStat](#)

## 24. Htop

htop is a ncurses-based process viewer. This is similar to top, but is more flexible and user friendly. You can interact with the htop using mouse. You can scroll vertically to view the full process list, and scroll horizontally to view the full command line of the process.

htop output consists of three sections 1) header 2) body and 3) footer.

**Header** displays the following three bars, and few vital system information. You can change any of these from the htop setup menu.

- CPU Usage: Displays the %used in text at the end of the bar. The bar itself will show different colors. Low-priority in blue, normal in green, kernel in red.
- Memory Usage
- Swap Usage

**Body** displays the list of processes sorted by %CPU usage. Use arrow keys, page up, page down key to scroll the processes.

**Footer** displays htop menu commands.

More info: [HTOP Screenshot and Examples](#)

## 25. Socket Statistics – SS

ss stands for socket statistics. This displays information that are similar to netstat command.

To display all listening sockets, do ss -l as shown below.

```
$ ss -l
```

Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
0	100	:::8009	:::*
0	128	:::sunrpc	:::*
0	100	:::webcache	:::*
0	128	:::ssh	:::*
0	64	:::nrpe	:::*

The following displays only the established connection.

```
$ ss -o state established
```

Recv-Q	Send-Q	Local Address:Port	Peer Address:Port	
0	52	192.168.1.10:ssh	<a href="#">192.168.2.11:55969</a>	timer:(on,414ms,0)

The following displays socket summary statistics. This displays the total number of sockets broken down by the type.

```
$ ss -s
```

```
Total: 688 (kernel 721)
```

```
TCP: 16 (estab 1, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 11
```

Transport	Total	IP	IPv6
*	721	-	-
RAW	0	0	0
UDP	13	10	3
TCP	16	7	9

INET	29	17	12
FRAG	0	0	0

<div class="gmail-m\_-154883216493872265gmail-m\_7491360818094342144gmail-lecture-container"