Sla

Hh

Call login

Daily checklist

**Disk Space:**

df -h

df -h

(Make sure you have enough disk space)

# ****Memory:****

free -m

(Check you're not out of memory)

# ****Processes / Load average****

top (shift + m)

htop

(Check for processes that are taking up a lot of memory/CPU)

# ****Apache errors****

cat /var/log/apache2/error.log

(Look for 500 errors caused by erroneous code on the server)

# ****High hit rate****

grep MaxClients /var/log/apache2/error.log

(Check for MaxClients warningdamn in your apache error logs)

tail -f /var/log/apache2/access.log

(Check for bots/spiders) [You might need to lower your MaxClients settings]

# ****Check recent logs****

ls -lrt /var/log/

(the -lrt flag will show you the most recently modified files at the end)

# ****Check your cronjobs****

ls -la /var/spool/cron/\*

ls -la /etc/cron\*

2049 nfs

Samba 137 /138

Linux rescue process

1. Need to boot from disk(CD or floppy)
2. Commands(fsck,fdisk,e2label,sync)

Fsck-file system check for file system checking and reapiring.

Fdisk-

E2label-changing label to a particular partition

ls /lib/modules/ karnal version

/sbin/mkbootdisk –device /dev/fd0 2.4.333

/mnt/sysimage do any change in system

If u like to make ur system the root env

Chroot /mnt/sysimage

All prev partition exist as well as other new partion boot created in /mnt/sysimage/

If a file system error it cant be mounted to enter into minimal rescue mode

For Check file system.

Fdisk –l

E2fsck /dev/sda1

E2label /dev/sda1 (boot)

Mkdir tmpmount

Mount /dev/sda2 /tmpmount

Chnge write down to disk(sync)

Skip then mont to boot n / to other directory

Mount it to that directory

Do change save n sync

Then umount

**cd /etc/sysconfig/network-scripts # cat route-eth2** GATEWAY0=10.10.12.1 NETMASK0=255.255.255.0 ADDRESS0=10.10.12.0 GATEWAY1=172.16.45.1 NETMASK1=255.255.255.0 ADDRESS1=172.16.45.1

squid default port 3128

error

icon

squid conf

http\_port 8080

cert

##local network allowed##

acl localnet src 172.25.5.0/24

acl gmail dstdomain .mail.gmail.com

acl gmail\_allow\_ip src "/etc/squid/gmail\_allow\_ip.acl"

acl youtube dstdomain .youtube.com

acl youtube\_allow\_ip src "/etc/squid/you\_allow\_ip.acl"

acl blockdownload url\_regex "/etc/squid/blockdownload\_ip.acl"

http\_access deny blcokdownd

http\_access deny youtube !youtube\_allow-ip

http\_access deny gmail !gmail\_allow-ip

http\_access allow localnet

http\_acees deny all

in block download

regular expression

.\[Ee][Xx][Ee]\*

## SELinux Modes

It's time to start playing around with SELinux, so let's begin with SELinux modes. At any one time, SELinux can be in any of three possible modes:

* Enforcing
* Permissive
* Disabled

In enforcing mode SELinux will enforce its policy on the Linux system and make sure any unauthorized access attempts by users and processes are denied. The access denials are also written to relevant log files. We will talk about SELinux policies and audit logs later.

Permissive mode is like a semi-enabled state. SELinux doesn't apply its policy in permissive mode, so no access is denied. However any policy violation is still logged in the audit logs. It's a great way to test SELinux before enforcing it.

The disabled mode is self-explanatory – the system won't be running with enhanced security.

### Checking SELinux Modes and Status

We can run the getenforce command to check the current SELinux mode.

getenforce

SELinux should currently be disabled, so the output will look like this:

Disabled

We can also run the sestatus command:

sestatus

When SELinux is disabled the output will show:

SELinux status: disabled

## SELinux Configuration File

The main configuration file for SELinux is /etc/selinux/config. We can run the following command to view its contents:

cat /etc/selinux/config

The output will look something like this:

# This file controls the state of SELinux on the system.

# SELINUX= can take one of these three values:

# enforcing - SELinux security policy is enforced.

# permissive - SELinux prints warnings instead of enforcing.

# disabled - No SELinux policy is loaded.

SELINUX=disabled

# SELINUXTYPE= can take one of these two values:

# targeted - Targeted processes are protected,

# minimum - Modification of targeted policy. Only selected processes are protected.

# mls - Multi Level Security protection.

SELINUXTYPE=targeted

There are two directives in this file. The SELINUX directive dictates the SELinux mode and it can have three possible values as we discussed before.

The SELINUXTYPE directive determines the policy that will be used. The default value is targeted. With a targeted policy, SELinux allows you to customize and fine tune access control permissions. The other possible value is "MLS" (multilevel security), an advanced mode of protection. Also with MLS, you need to install an additional package.

## Resize /tmp Partition

Many of you might have come across situations where your /tmp gets filled very often either due to eaccelerator cache or due to session files or such temporary files. Cpanel by default creates /tmp with 512M size. You can always resize tmp to your choice. Here is the step by step details on how it can be done.

1) Stop mysql, apache and cpanel to prevent writing temporary files to /tmp.

/etc/init.d/chkservd stop  
/etc/rc.d/init.d/mysql stop  
/etc/rc.d/init.d/httpd stop  
/etc/rc.d/init.d/cpanel stop

2) Unmount tmp partition.

umount /var/tmp  
umount /tmp

Sometimes you will receive errors stating that the device is busy or /tmp cannot be unmounted. Then find out all processes using /tmp and kill them.

lsof /tmp

The above command will list the process ids currently using /tmp. Kill the pids as follows.

kill -9 pid

eg: kill -9 3766

3) The cpanel /scripts/securetmp is the one which maintains the tmp size. You can edit the following line in it to change the size.( In /usr/local/cpanel/scripts/securetmp)

my $tmpdsksize = 512000;

Suppose you want to raise the partition size to 2G, you can also do it as follows.

sed -i -e ‘s/512000/2048000/g’ /scripts/securetmp

The above will replace 512000 with 2048000 in the file /scripts/securetmp

4) Remove the temp disk

rm /usr/tmpDSK

5) Now, run the following to recreate tmp

/scripts/securetmp –auto

6) Now go to /tmp and set the mysql socket file.

cd /tmp  
ln -s /var/lib/mysql/mysql.sock

7) Restart all services

/etc/rc.d/init.d/mysql start  
/etc/rc.d/init.d/httpd start  
/etc/rc.d/init.d/cpanel start  
/etc/init.d/chkservd start

8) Now check your tmp size using the command df -h

umount -l /tmp

umount -l /var/tmp

rm -fv /usr/tmpDSK

/usr/local/cpanel/scripts/securetmp

https://admin-ahead.com/forum/Themes/default/images/post/xx.gif

##### [Increase /tmp partition size](https://admin-ahead.com/forum/general-linux/increase-tmp-partition-size/msg266/#msg266)

« **on:** November 05, 2013, 10:56:46 pm »

How to increase /tmp partition size  
===  
If you need to increase the disk space of the partition for some reason, you will have to create a virtual partition and have to mount it on /tmp.  
  
The following steps will guide you to create a virtual partition:  
  
1) To create a partition of 2GB, use the below dd command:  
  
dd if=/dev/zero of=/home/tmp-dir bs=1024M count=2  
  
2) Once the partition is created, you need to create the file system on it using the mke2fs command  
  
mke2fs -j /home/tmp-dir  
  
3) Now, the partition is ready to be used but you need to mount it on /tmp directory.  
  
mount -t ext3 -o loop /home/tmp-dir /tmp  
  
Here, we have used ‘loop’ while mounting /home/tmp-dir partition because we are not mounting an actual block device but to make a file accessible as a block device.  
  
4) To verify the partition, execute  
  
mount  
  
5) To make sure this partition is mounted automatically after every reboot, edit the /etc/fstab file and replace the /tmp line with the following one:  
  
/home/tmp-dir /tmp ext3 defaults,loop 0 0  
  
===  
Hope, this helps.

**Dynamic DNS** (**DDNS** or **DynDNS**) is a method of automatically updating a name server in the Domain Name System (DNS), often in Not Real Time, with the active **DDNS** configuration of its configured hostnames, addresses or other information. The term is used to describe two different concepts.

System monitoring

20 Command Line Tools to Monitor Linux Performance

BY [RAVI SAIVE](http://www.tecmint.com/author/admin/) LAST UPDATED: JANUARY 3, 2015

 Download Your Free eBooks NOW - [10 Free Linux eBooks for Administrators](http://www.tecmint.com/10-useful-free-linux-ebooks-for-newbies-and-administrators/" \t "_blank) | [4 Free Shell Scripting eBooks](http://www.tecmint.com/free-linux-shell-scripting-books/" \t "_blank)

It’s really very tough job for every System or Network administrator to monitor and debug Linux System Performance problems every day. After being a Linux Administrator for 5 years in IT industry, I came to know that how hard is to monitor and keep systems up and running. For this reason, we’ve compiled the list of Top 20frequently used command line monitoring tools that might be useful for every Linux/Unix System Administrator. These commands are available under all flavors of Linux and can be useful to monitor and find the actual causes of performance problem. This list of commands shown here are very enough for you to pick the one that is suitable for your monitoring scenario.

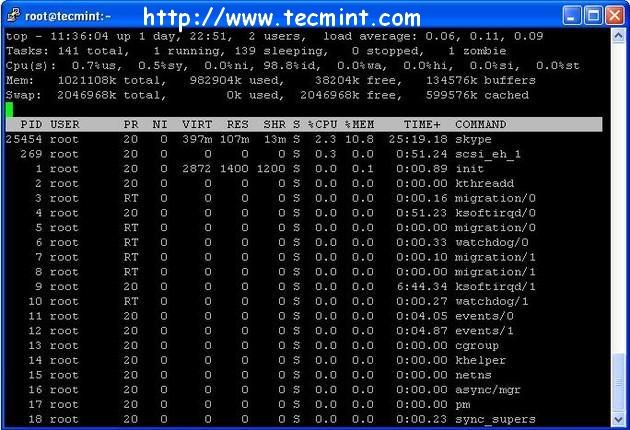
[](http://www.tecmint.com/command-line-tools-to-monitor-linux-performance/linux-command-line-monitoring-tools/)

*Linux Command Line Monitoring*

1. Top – Linux Process Monitoring

Linux Top command is a performance monitoring program which is used frequently by many system administrators to monitor Linux performance and it is available under many Linux/Unix like operating systems. The top command used to dipslay all the running and active real-time processes in ordered list and updates it regularly. It display CPU usage, Memory usage, Swap Memory, Cache Size, Buffer Size, Process PID, User,Commands and much more. It also shows high memory and cpu utilization of a running processess. The top command is much userful for system administrator to monitor and take correct action when required. Let’s see top command in action.

# top

[](http://www.tecmint.com/command-line-tools-to-monitor-linux-performance/Top%20Command%20Example)

*Top Command Example*

For more examples of Top command read : [12 TOP Command Examples in Linux](http://www.tecmint.com/12-top-command-examples-in-linux/" \t "_blank)

2. VmStat – Virtual Memory Statistics

Linux VmStat command used to display statistics of virtual memory, kernerl threads, disks, system processes,I/O blocks, interrupts, CPU activity and much more. By default vmstat command is not available under Linux systems you need to install a package called sysstat that includes a vmstat program. The common usage of command format is.

**# vmstat**

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

r b swpd free inact active si so bi bo in cs us sy id wa st

1 0 0 810420 97380 70628 0 0 115 4 89 79 1 6 90 3 0

For more Vmstat examples read : [6 Vmstat Command Examples in Linux](http://www.tecmint.com/linux-performance-monitoring-with-vmstat-and-iostat-commands/" \t "_blank)

3. Lsof – List Open Files

Lsof command used in many Linux/Unix like system that is used to display list of all the open files and the processes. The open files included are disk files, network sockets, pipes, devices and processes. One of the main reason for using this command is when a disk cannot be unmounted and displays the error that files are being used or opened. With this commmand you can easily identify which files are in use. The most common format for this command is.

**# lsof**

COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME

init 1 root cwd DIR 104,2 4096 2 /

init 1 root rtd DIR 104,2 4096 2 /

init 1 root txt REG 104,2 38652 17710339 /sbin/init

init 1 root mem REG 104,2 129900 196453 /lib/ld-2.5.so

init 1 root mem REG 104,2 1693812 196454 /lib/libc-2.5.so

init 1 root mem REG 104,2 20668 196479 /lib/libdl-2.5.so

init 1 root mem REG 104,2 245376 196419 /lib/libsepol.so.1

init 1 root mem REG 104,2 93508 196431 /lib/libselinux.so.1

init 1 root 10u FIFO 0,17 953 /dev/initctl

More lsof command usage and examples : [10 lsof Command Examples in Linux](http://www.tecmint.com/10-lsof-command-examples-in-linux/" \t "_blank)

4. Tcpdump – Network Packet Analyzer

Tcpdump one of the most widely used command-line network packet analyzer or packets sniffer program that is used capture or filter TCP/IP packets that received or transferred on a specific interface over a network. It also provides a option to save captured packages in a file for later analysis. tcpdump is almost available in all major Linux distributions.

**# tcpdump -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

22:08:59.617628 IP tecmint.com.ssh > 115.113.134.3.static-mumbai.vsnl.net.in.28472: P 2532133365:2532133481(116) ack 3561562349 win 9648

22:09:07.653466 IP tecmint.com.ssh > 115.113.134.3.static-mumbai.vsnl.net.in.28472: P 116:232(116) ack 1 win 9648

22:08:59.617916 IP 115.113.134.3.static-mumbai.vsnl.net.in.28472 > tecmint.com.ssh: . ack 116 win 64347

For more tcpdump usage read : [12 Tcpdump Command Examples in Linux](http://www.tecmint.com/12-tcpdump-commands-a-network-sniffer-tool/" \t "_blank)

5. Netstat – Network Statistics

Netstat is a command line tool for monitoring incoming and outgoing network packets statistics as well as interface statistics. It is very useful tool for every system administrator to monitor network performance and troubleshoot network related problems.

**# netstat -a | more**

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 \*:mysql \*:\* LISTEN

tcp 0 0 \*:sunrpc \*:\* LISTEN

tcp 0 0 \*:realm-rusd \*:\* LISTEN

tcp 0 0 \*:ftp \*:\* LISTEN

tcp 0 0 localhost.localdomain:ipp \*:\* LISTEN

tcp 0 0 localhost.localdomain:smtp \*:\* LISTEN

tcp 0 0 localhost.localdomain:smtp localhost.localdomain:42709 TIME\_WAIT

tcp 0 0 localhost.localdomain:smtp localhost.localdomain:42710 TIME\_WAIT

tcp 0 0 \*:http \*:\* LISTEN

tcp 0 0 \*:ssh \*:\* LISTEN

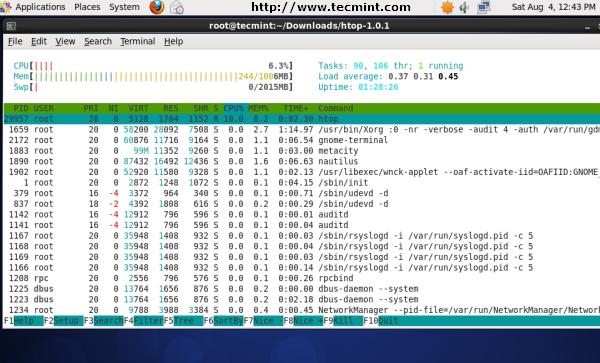
tcp 0 0 \*:https \*:\* LISTEN

More Netstat examples : [20 Netstat Command Examples in Linux](http://www.tecmint.com/20-netstat-commands-for-linux-network-management/).

6. Htop – Linux Process Monitoring

Htop is a much advanced interactive and real time Linux process monitoring tool. This is much similar to Linuxtop command but it has some rich features like user friendly interface to manage process, shortcut keys,vertical and horizontal view of the processes and much more. Htop is a third party tool and doesn’t included in Linux systems, you need to install it using YUM package manager tool. For more information on installation read our article below.

**# htop**



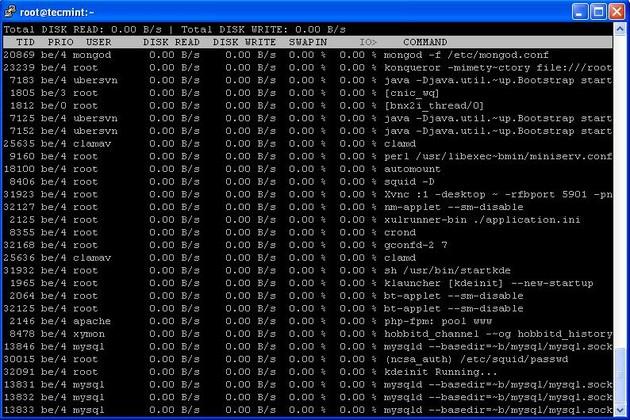
*Htop Command Example Screenshot*

For Htop installation read : [Install Htop (Linux Process Monitoring) in Linux](http://www.tecmint.com/install-htop-linux-process-monitoring-for-rhel-centos-fedora/" \t "_blank)

7. Iotop – Monitor Linux Disk I/O

Iotop is also much similar to top command and Htop program, but it has accounting function to monitor and display real time Disk I/O and processes. This tool is much useful for finding the exact process and high used disk read/writes of the processes.

**# iotop**



*Iotop Command Example Screenshot*

For Ioptop installation and usage read : [Install Iotop in Linux](http://www.tecmint.com/install-iotop-monitor-linux-disk-io-in-rhel-centos-and-fedora/" \t "_blank)

8. Iostat – Input/Output Statistics

IoStat is simple tool that will collect and show system input and output storage device statistics. This tool is often used to trace storage device performance issues including devices, local disks, remote disks such asNFS.

**# iostat**

Linux 2.6.18-238.9.1.el5 (tecmint.com) 09/13/2012

avg-cpu: %user %nice %system %iowait %steal %idle

2.60 3.65 1.04 4.29 0.00 88.42

Device: tps Blk\_read/s Blk\_wrtn/s Blk\_read Blk\_wrtn

cciss/c0d0 17.79 545.80 256.52 855159769 401914750

cciss/c0d0p1 0.00 0.00 0.00 5459 3518

cciss/c0d0p2 16.45 533.97 245.18 836631746 384153384

cciss/c0d0p3 0.63 5.58 3.97 8737650 6215544

cciss/c0d0p4 0.00 0.00 0.00 8 0

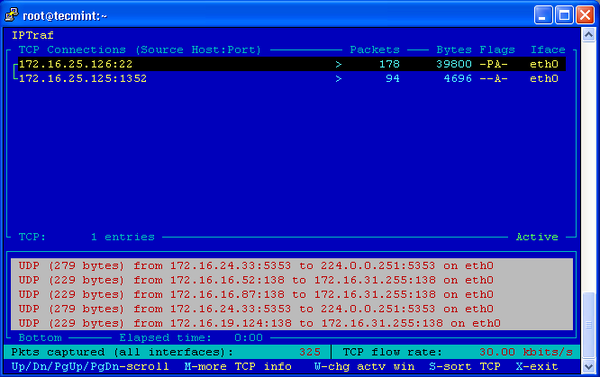
cciss/c0d0p5 0.63 3.79 5.03 5936778 7882528

cciss/c0d0p6 0.08 2.46 2.34 3847771 3659776

For more Iostat usage and examples visit : [6 Iostat Command Examples in Linux](http://www.tecmint.com/linux-performance-monitoring-with-vmstat-and-iostat-commands/" \t "_blank)

9. IPTraf – Real Time IP LAN Monitoring

IPTraf is an open source console-based real time network (IP LAN) monitoring utility for Linux. It collects a variety of information such as IP traffic monitor that passes over the network, including TCP flag information, ICMP details, TCP/UDP traffic breakdowns, TCP connection packet and byne counts. It also gathers information of general and detaled interface statistics of TCP, UDP, IP, ICMP, non-IP, IP checksum errors, interface activity etc.

[](http://www.tecmint.com/wp-content/uploads/2013/02/IPTraf2.png)

*IP Traffic Monitor*

For more information and usage of IPTraf tool, please visit : [IPTraf Network Monitoring Tool](http://www.tecmint.com/real-time-interactive-ip-lan-monitoring-with-iptraf-tool/" \t "_blank)

10. Psacct or Acct – Monitor User Activity

psacct or acct tools are very useful for monitoring each users activity on the system. Both daemons runs in the background and keeps a close watch on the overall activity of each user on the system and also what resources are being consumed by them.

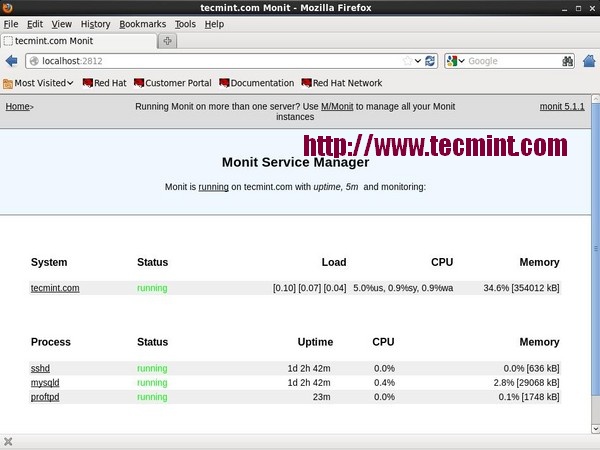
These tools are very useful for system administrators to track each users activity like what they are doing, what commands they issued, how much resources are used by them, how long they are active on the system etc.

For installation and example usage of commands read the article on [Monitor User Activity with psacct or acct](http://www.tecmint.com/how-to-monitor-user-activity-with-psacct-or-acct-tools/" \t "_blank)

11. Monit – Linux Process and Services Monitoring

Monit is a free open source and web based process supervision utility that automatically monitors and managers system processes, programs, files, directories, permissions, checksums and filesystems.

It monitors services like Apache, MySQL, Mail, FTP, ProFTP, Nginx, SSH and so on. The system status can be viewed from the command line or using it own web interface.

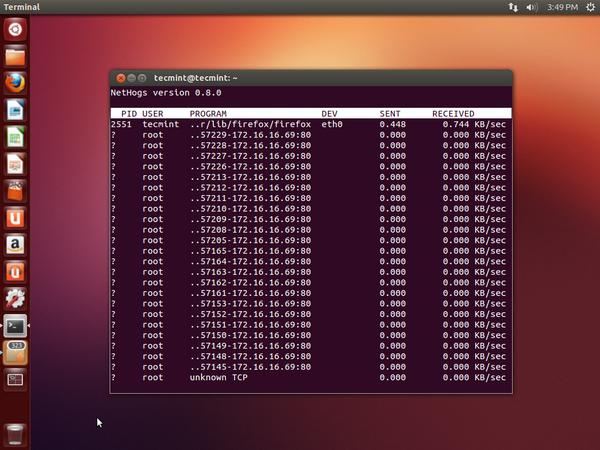
[](http://www.tecmint.com/wp-content/uploads/2013/02/Monit-Monitoring.jpg)

*Monit Linux Process Monitoring*

Read More : [Linux Process Monitoring with Monit](http://www.tecmint.com/how-to-install-and-setup-monit-linux-process-and-services-monitoring-program/" \t "_blank)

12. NetHogs – Monitor Per Process Network Bandwidth

NetHogs is an open source nice small program (similar to Linux top command) that keeps a tab on each process network activity on your system. It also keeps a track of real time network traffic bandwidth used by each program or application.

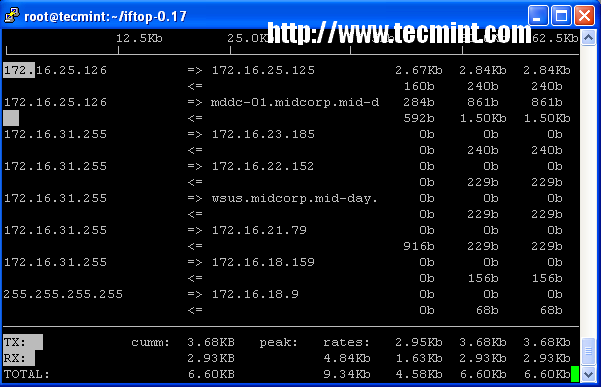
[](http://www.tecmint.com/wp-content/uploads/2013/02/NetHogs-bandwidth-Monitoring.jpg)

*NetHogs Linux Bandwidth Monitoring*

Read More : [Monitor Linux Network Bandwidth Using NetHogs](http://www.tecmint.com/nethogs-monitor-per-process-network-bandwidth-usage-in-real-time/" \t "_blank)

13. iftop – Network Bandwidth Monitoring

iftop is another terminal-based free open source system monitoring utility that displays a frequently updated list of network bandwidth utilization (source and destination hosts) that passing through the network interface on your system. iftop is considered for network usage, what ‘top‘ does for CPU usage. iftop is a ‘top‘ family tool that monitor a selected interface and displays a current bandwidth usage between two hosts.

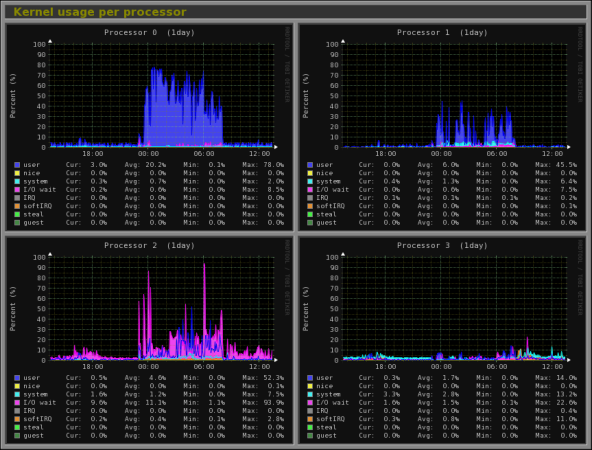
[](http://www.tecmint.com/wp-content/uploads/2013/04/iftop.png)

*iftop – Network Bandwidth Monitoring*

Read More : [iftop – Monitor Network Bandwidth Utilization](http://www.tecmint.com/install-iftop-bandwidth-monitoring-tool-in-rhel-centos-fedora/" \t "_blank)

14. Monitorix – System and Network Monitoring

Monitorix is a free lightweight utility that is designed to run and monitor system and network resources as many as possible in Linux/Unix servers. It has a built in HTTP web server that regularly collects system and network information and display them in graphs. It Monitors system load average and usage, memory allocation, disk driver health, system services, network ports, mail statistics (Sendmail, Postfix, Dovecot, etc), MySQL statistics and many more. It designed to monitor overall system performance and helps in detecting failures, bottlenecks, abnormal activities etc.

[](http://www.tecmint.com/command-line-tools-to-monitor-linux-performance/monitorix-monitoring/)

*Monitorix Monitoring*

Read More : [Monitorix a System and Network Monitoring Tool for Linux](http://www.tecmint.com/monitorix-a-lightweight-system-and-network-monitoring-tool-for-linux/" \t "_blank)

15. Arpwatch – Ethernet Activity Monitor

Arpwatch is a kind of program that is designed to monitor Address Resolution (MAC and IP address changes) ofEthernet network traffic on a Linux network. It continuously keeps watch on Ethernet traffic and produces a log of IP and MAC address pair changes along with a timestamps on a network. It also has a feature to send an email alerts to administrator, when a pairing added or changes. It is very useful in detecting ARP spoofing on a network.

Read More : [Arpwatch to Monitor Ethernet Activity](http://www.tecmint.com/monitor-ethernet-activity-in-linux/" \t "_blank)

16. Suricata – Network Security Monitoring

Suricata is an high performance open source Network Security and Intrusion Detection and Prevention Monitoring System for Linux, FreeBSD and Windows.It was designed and owned by a non-profit foundationOISF (Open Information Security Foundation).

Read More : [Suricata – A Network Intrusion Detection and Prevention System](http://www.tecmint.com/suricata-a-network-intrusion-detection-prevention-system/" \t "_blank)

17. VnStat PHP – Monitoring Network Bandwidth

VnStat PHP a web based frontend application for most popular networking tool called “vnstat“. VnStat PHPmonitors a network traffic usage in nicely graphical mode. It displays a total IN and OUT network traffic usage inhourly, daily, monthly and full summary report.

Read More : [VnStat PHP – Monitoring Network Bandwidth](http://www.tecmint.com/vnstat-php-frontend-for-monitoring-network-bandwidth/" \t "_blank)

18. Nagios – Network/Server Monitoring

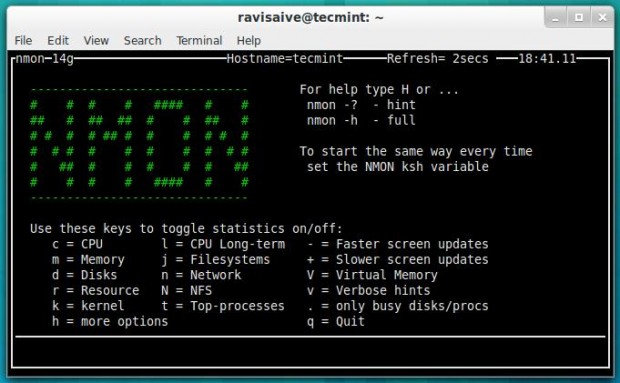
Nagios is an leading open source powerful monitoring system that enables network/system administrators to identify and resolve server related problems before they affect major business processes. With the Nagios system, administrators can able to monitor remote Linux, Windows, Switches, Routers and Printers on a single window. It shows critical warnings and indicates if something went wrong in your network/server which indirectly helps you to begin remediation processes before they occur.

Read More : [Install Nagios Monitoring System to Monitor Remote Linux/Windows Hosts](http://www.tecmint.com/install-nagios-in-linux/" \t "_blank)

19. Nmon: Monitor Linux Performance

Nmon (stands for Nigel’s performance Monitor) tool, which is used to monitor all Linux resources such as CPU, Memory, Disk Usage, Network, Top processes, NFS, Kernel and much more. This tool comes in two modes: Online Mode and Capture Mode.

The Online Mode, is used for real-time monitoring and Capture Mode, is used to store the output in CSV format for later processing.

[](http://www.tecmint.com/wp-content/uploads/2013/12/Nmon.jpeg)

*Nmon Monitoring*

Read More: [Install Nmon (Performance Monitoring) Tool in Linux](http://www.tecmint.com/nmon-analyze-and-monitor-linux-system-performance/" \t "_blank)

20. Collectl: All-in-One Performance Monitoring Tool

Collectl is a yet another powerful and feature rich command line based utility, that can be used to gather information about Linux system resources such as CPU usage, memory, network, inodes, processes, nfs, tcp, sockets and much more.

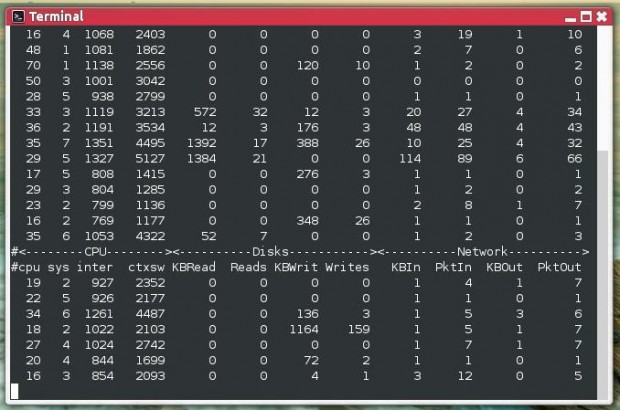
Patch management

Yum,rpm

Monitoring Disk space Utilization- df –h

File system creation / mounting / formatting

Taking regular and carrying out restoration when required cronjob tar

[](http://www.tecmint.com/wp-content/uploads/2013/12/Collectl.jpg)

*Collectl Monitoring*

Read More: [Install Collectl (All-in-One Performance Monitoring) Tool in Linux](http://www.tecmint.com/linux-performance-monitoring-with-collectl-tool/" \t "_blank)

Start/Stop the in-hoapplications-init start service

Should have Netbackup knowledge- Symantec nt backup

Samba / NFS / Apache configuration

Other interview question

**10. Summarizing, what are all the differences between soft and hard links?**  
Soft Links:

* Soft Links can be created across file systems.
* Soft link has a different inode number than the original file.
* On deleting the original file, soft link cannot be accessed.
* Soft link needs extra memory to store the original file name as its data.
* Source file need not exist for soft link creation.
* Can be created on a file or on a directory.
* Access to the file is slower due to the overhead to access file.

Hard Links:

* Hard links can be created only within the file system.
* Hard links have the same inode number as the original file.
* On deleting the original file, hard linked file can still be accessed.
* Hard links do not need any extra data memory to save since it uses links
* Source file should exist.
* Can be created only on files, not on directories.
* Access  to the file is faster compared to soft link.

- See more at: <http://www.theunixschool.com/2012/03/soft-links-hard-links-all-about-inodes.html#sthash.7ML3d1RZ.dpuf>

### Question 9 : What is HARD and SOFT mount in NFS ?

**In HARD mount ...**  
  
If the NFS file system is hard mounted, the NFS daemons will try repeatedly to contact the server. The NFS daemon retries will not time out, will affect system performance, and you cannot interrupt them  
  
If you just mount a file system without specifying hard or soft, the default is a hard mount. Hard mounts are preferable because of the stateless nature of NFS.  
  
If a client sends an I/O request to the server (such as an ls -la), and the server gets rebooted, in client machine the process will keep on running.  
This preserves data transfers in the event of a server failure  
  
**IN SOFT Mount :**  
  
A soft mount allows the client to stop trying an operation after a period of time. If the NFS server goes down at the time of data transfer , it will alert and the process will do down.Thsi may cause the data corruption.  
A soft link will return with an error and fail.  
  
you should only use soft mounts in the cases where client responsiveness is more important than data integrity.  
  
In another word ..soft mount will allow automatic unmount if the filesystem is idle for a specified time period 

### Question 10 : Explain NFS mount options ?

Syntax to mount NFS FS:  
  
#mount -t vfstype [-o options] NFS Servername:/exporteddirectory /mount point  
  
or  
  
#mount -t nfs -o options host:/remote/export /local/directory  
  
  
  
**Mount options explained below :**  
  
1. -0 initr  
  
This option is used in non reliable network, or network having more network congestion. NFS request will be interrupted when server is not reachable.  
  
2. -o hard   
  
If hard option is specified during nfs mount, user cannot terminate the process waiting for NFS communication to resume. For ex ..if u ran ls -a command on ur NFS mounted directory but that time ur NFS server went down means .  
The process wont get killed or stopped ..it will wait until the NFS server and mount poit become available.  
  
3. -o soft   
  
If soft option is specified during nfs mount, user will get error alert when NFS server is not reachable. This is just inverse of hard mount option. It wont wait for reply if the NFS server went down , it will alert us and the process will go down.  
  
4. -o Nfsvers=value  
  
If this option is specified during nfs mount NFS client uses particular NFS protocol version to communicate.   
  
For example - TCP   
  
# mount -t nfs -o tcp 192.168.1.4:/mnt/array1/RHEL5 /data/  
# mount | grep -i tcp  
 192.168.1.4:/mnt/array1/RHEL5 on /data type nfs (rw,tcp,addr=192.168.1.4)  
  
The Difference between HARD and SOFT mount option explained in another POST. 

In accelay kale interview

How to know the current runlevel

Ans-runlevel, who -r

How to change runlevel

Ans-vi /etc/initab and

change default initdefault value

analysis of system monitoring tools

On Unix and Unix-like computer operating systems, a **zombie process**or **defunct process** is a **process** that has completed execution (via the exit system call) but still has an entry in the **process** table: it is a**process** in the "Terminated state".

Its name is kind of unique for something that shows system statistics: top. It is a part of the procps package, a set of Linux utilities that provide system information. Besides top, procps also includes free, vmstat, ps, and many other tools.

Top gives you a snapshot of a system's situation, e.g., free physical memory, number of running tasks, percentage of CPU time usage of each processes--all in a single view. So it's like using ps, free, and uptime at the same time. Top gets most of its information from several files under the /proc directory. You might already be aware that this directory provides users with a broad range of system-wide statistics, but thanks to top, we can summarize most of them inside a single organized window.

Furthermore, with top you can do things like:

* Sort processes based on CPU time usage, memory usage, total runtime, etc.
* Switching to multi view mode. In this mode, you're given up to four windows and each window can be assigned different settings (sorting, coloring, displayed field)
* Send a signal by mentioning the target process' PID and the signal number.

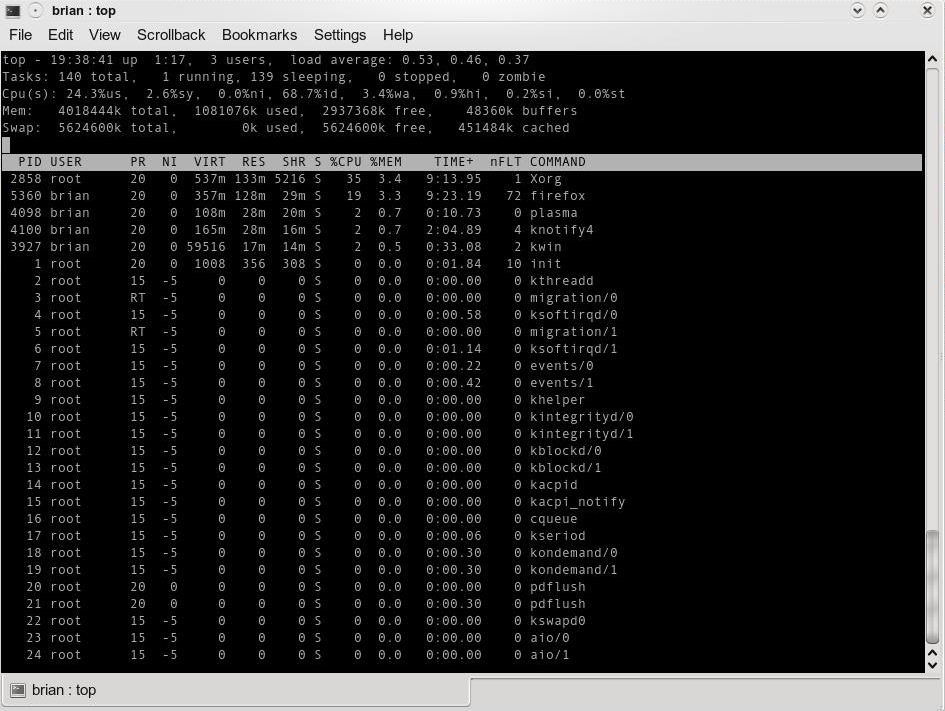
To learn how to use top more effectively, please refer to [this article](http://www.linuxforums.org/articles/using-top-more-efficiently_89.html).

However, top is not the ideal tool for gathering system statistics in long intervals, say, a week or a month. sadc (system activity data collector) is a better tool for this kind of interval. sadc generates a log that contains system statistics and uses sar to generate report and summaries based on this log. Some people use top in batch mode as an alternative, but it's not as effective as sadc because:

* sadc records the data in binary format, thus requiring smaller space
* according to personal experiment, by using "time" command, sadc works faster than top therefore the tool itself doesn't introduce big overhead

That's enough to reintroduce top. Let's begin by understanding the meaning of each part of top's "dashboard." Also, throughout this article, it's important to note the meaning of "high" or "low." There is no absolute threshold to precisely label a number high or low. I tend to apply the high or low label if the statistic is higher or lower compared to the average number you usually see in the related field.

## Processor Statistics

****Figure 1

On the uppermost line, shown in Figure 1, there are (from left to right): current time (hour:minute:second), uptime (hour:minute), number of active user IDs, and load average.

The first two are self-explanatory. How about the number of active user IDs? One might think it's the number of currently logged-in users, either interactively or not (via ssh, for example). Not exactly. It also counts the number of pseudo terminals spawned under certain user ID privileges. Let's say you login into GNOME as "joe" and then you spawn two xterm windows, then it counts you as three users. Top increments the number by one to account the existence of init process.

Load average is the representation of the number of processes marked as runnable in the run queue. From left to right, they represent one-minute, five-minute, and 15-minute load averages. They are the same three numbers you can see in /proc/loadavg. A detailed explanation can be found in this [Linux Journal](http://www.linuxjournal.com/article/9001)article. Bear in mind that a process in a runnable state doesn't neccessarily mean it's currently executed by the processor. It's a mark stating that "I (the task) am ready to be executed, but it's up to the scheduler to decide when to pick me up."

For example, if you have a single dual-core processor and one-minute load average reads "2," then on average each core is executing one process. If it reads "4," then on average one process is being executed and one is waiting in each core. The actual situation might be different, since load average is actually showing "normalized trends" during certain intervals instead of discrete numbers at any given moment.

Moving one line below, we see statistics about number of tasks in various states:

* Running: as we have discussed before, this represents the number of tasks in a runnable state.
* Sleeping: processes being blocked due to waiting for an event (e.g., time out or I/O completion). It accounts for both interruptible (can be awaken earlier by SysV signal) or uninterruptible (completely ignoring SysV signal) processes.
* Stopped: the exact meaning here is "paused," not "terminated." In a terminal, you can stop a program by sending it a SIGSTOP signal or pressing Ctrl-Z if it's a foreground task.
* Zombie: "A dead body without soul" might be a good analogy. After a child task is terminated, it is cleaned up and the only thing left is a task descriptor that includes a very important value: exit status. So if the number of zombies is high, that is a sign that one or more programs have a bug properly terminating child tasks.

Now we enter CPU section statistics. By pressing "1," you could switch between cumulative and per-core/physical CPU usage. The numbers here are derived from /proc/stat. Top does the sampling and converts them from counter-based number into percentages.

The two left-most fields, %us and %sy, represent percentage of CPU time spent in user mode and kernel mode, respectively.

What are user and kernel modes? Let's step back a bit and discuss the Intel x86 platform. In this case, the processor has four levels of privileges, ranging from 0 to 3, each called a "ring." Ring 0 is the most privileged domain--any codes executed when the CPU enters this level can access anything on the system. Ring 3 is the least-privileged domain.

The Linux kernel, by default, doesn't use rings 1 or 2, except in certain conditions i.e., entering a hypervisor. Ring 0 is also known as kernel mode, because almost all of kernel codes are granted this privilege. Ring 3 is, easy to guess, known as user mode. User space codes live in this ring.

This logical separation primarily serves as protection. User codes can do some work, but to touch underlying layers, such as reading or writing to the hard disk or allocating several kilobytes of RAM, the codes need to ask for kernel services, also known as system calls. System calls are composed of one or more kernel functions. They are running on behalf of the calling task. So from the privilege escalation point of view, it is the same task but switching from user mode to kernel mode. When a system call finishes, it returns a value and the task is moved back to user mode.

In my opinion, a low or high number on both of these mode values is not really a big concern. We can consider it a way to identify our system's characteristics. If you find the %sy higher than %us most of the time, it is likely that your machine serves as a file server, database server, or something similar. On the other hand, a higher %us means your machine does a lot of number crunching.

High %sy values could also mean some kernel threads are busy doing something. Kernel threads are like normal tasks, but they have a fundamental difference: they operate entirely in kernel mode. Kernel threads are created to do a specific job.

For example: pdflush writes back updated page caches to the related storage, and a migration thread does the load balancing between CPUs by reassigning some tasks to less-loaded processors. If kernel threads spend too much time in kernel mode, user space tasks will have less of a chance to run. Even with the latest scheduler like the [Completely Fair Scheduler](http://en.wikipedia.org/wiki/Completely_Fair_Scheduler) (CFS), which provides better fairness, you should read the related documentation to fine-tune the kernel thread.

%hi and %si are statistics related to CPU servicing interrupts. %hi is the percentage of CPU time servicing hard interrupts, while %si is when CPU denotes the soft interrupt time percentage.

Okay, so why are they called "soft" and "hard"? A little background on how Linux kernel handles interrupt first: when an interrupt arrives on a certain IRQ line, the Linux kernel quickly acknowledges it. The related handler in the IDT (Interrupt Descriptor Table) is executed. This acknowledgment process is done while disabling interrupt delivery in the local processor. This is why it has to be quick; if not, the number of pending interrupts will rise and will make hardware response worse.

The solution? Acknowledge fast, prepare some data, and then defer the real work until later. This deferred job is called a soft interrupt, because it's not really an interrupt but treated like one. By dividing the work of interrupt handling, system response gets smoother and indirectly gives user space code more time to run. Why? Simple--the interrupt handler runs in kernel mode, just like when we perform a system call.

A high %hi value means one or more devices are too busy doing their work, and are most likely overloaded. In certain cases, it might mean the device is broken, so it's good to do a thorough checking before it gets worse. Check /proc/interrupts and you can pinpoint the source.

Regarding %si, a high number here doesn't always related to high frequency of real interrupts. Dividing work between the real interrupt handler and soft IRQs is the driver's job, so high %si tends to show there's something needed to optimize inside the driver. The easiest way to optimize is upgrade your kernel. If you compile the kernel on your own, use the configuration file of the old kernel as the base config of your new kernel, so you can compare the result of soft IRQ management. Any third-party drivers should also be upgraded. A raw observation using Google tells that network drivers are usually the source of the problem.

## Memory Statistics

Let's move on virtual memory statistics. This area is derived from /proc/meminfo content.

What's "total memory" in your opinion? The size of your RAM? Not really. The real meaning is the total size of RAM that is mappable by your running kernel. If you're not using a kernel that has large addressing capability, the maximum addressable memory is around 896 Mb. 896 Mb is 1 Gb minus some reserved kernel-address space. Theoretically, 1 Gb is kernel address space, while 3 Gb is user-space address space in a 32-bit x86 architecture.

To make sure your system can do large addressing, install a kernel package that contains strings such as "PAE," "bigmem," or "hugemem." Using this modified kernel, you can map up to 64 Gb using feature called PAE (Physical Address Extension).

The two confusing fields in memory statistics are "buffers" and "cached." Cached? What's cache? First, one might get the wrong conclusion by its placement. Notice it is placed in "Swap" line, so one might take conclusion it is cache of swap or something similar. Not at all. It is actually part of "Mem" line, which describes physical memory consumption.

Buffer is the in-memory copy of blocks as a result of the kernel performing raw disk access. For example, when kernel wants to read content of a file, first the kernel has to retrieve the information from the related inode. To read this inode, the kernel reads a block from the block device at a certain sector offset. Therefore, we can conclude that buffer size rises if there is access to inode or dentry (directory entry), or superblock (when we mount a partition), when directly accessing a file or block device.

The cached field tells us the size of RAM used to cache the content of files being read recently. How is that different from buffer field? Recall that buffer size increases when we bypass the filesystem layer, and cached size increases when do the opposite. We can also see cached size as the degree of caching aggressiveness from the kernel. Both buffer and cache usually grow as we do more read operations--and this is perfectly normal. Subsequent reads could be satisfied by reading from cache, which enhances speed while reducing read latency.

## Per-process Statistics

The rest of the top's dashboard is per-process statistics. This is a very wide subtopic to cover, so choice must be made:

* Uncover the meaning of memory related fields: VIRT, RES, SHR, nFLT, and nDRT
* Understanding NI and PR fields and the connection between the two

"How much memory is allocated by task X?" you ask. You're confused between looking at VIRT, RES, or SHR. Before answering that, I'll take a quick recap about two types of memory area: anonymous or file-backed.

An example of an anonymous memory area is the result of executing malloc(), a C function for requesting certain amount of memory. It might happen when your word processor allocates a buffer to hold the characters you have typed. Or when a web browser prepares chunks of memory to display a website.

What about file-backed memory area? When your program loads a shared library, a program called loader loads and maps the requested library into process's address space. Thus, from the program's point of view, accessing the library (i.e., calling a function) is like pointing to certain memory address. Refer to this[LinuxDevCenter article](http://linuxdevcenter.com/pub/a/linux/2006/11/30/linux-out-of-memory.html) for further information about memory allocation.

VIRT refers to the length of memory area, while RES shows us how many memory blocks (called pages) are really allocated and mapped to process address space. Thus, VIRT is like talking about the size of the land we own, but the house, what RES represents, built on top of it doesn't necessarily have to occupy the entire space. Quite likely, RES is far smaller than VIRT and it's perfectly logical because of the demands of the paging mechanism, which only allocate pages when it is really needed. So, the answer is clear: RES shows you the approximation of per-process memory consumption.

But RES itself has a flaw. Recall that we include the size of file-backed memory area in the calculation (e.g., for the shared library). We know that two or more running programs might use same set of libraries. Think about KDE- or GNOME-based programs. When you run those programs, they will refer to the same libraries but mapped to their own address space. You might get the wrong conclusion those programs require a big area, while in reality the sum of pages that belong to the running programs are lower than the sum of their RES.

This is where SHR comes to rescue. SHR shows us the size of the file-backed memory area. Thus, RES minus SHR leaves us with the size of anonymous memory area. So, actual memory consumption of a process is somewhere between RES and RES minus SHR, depending on how many shared libraries are loaded by any given process and other processes using those same libraries.

Let's move to other fields: nFLT and nDRT. These two fields don't show up by default, so you need to select them first by pressing 'f' or 'o' and pressing the key representing the field. Recall that the kernel does demand paging if a process points to valid memory address but no page is present there. The formal name of this condition is page fault. Page fault has two types: minor and major fault. Minor page faults happen if no storage access is involved. On the other hand, major page faults do involve storage access.

A minor page fault is connected with the anonymous memory area, because there is no need to read the disk. The exception to this is when the pages are swapped out. A major page fault is connected to the file-backed memory area. On every major fault, disk blocks are read and brought to RAM. However, thanks to page cache, if the target disk block's content is found there then it will be a minor fault.

nFLT shown by top only counts the number of major page faults. If this number gets high, it is a good indication that you need more RAM. If free memory gets tight, some pages in page cache will be flushed out and anonymous memory will be swapped out. Thus, if they are accessed, additional major faults occur. Try to add more RAM, until you see moderate value for nFLT. Also check your program's internal log or statistics and see if high nFLT means reduced performance. Usually, latency has a tight relationship with major page faults.

nDRT is supposed to display the number of dirty pages. But without a clear reason, this value (which is taken from rightmost field of /proc/<pid>/statm) always shows zero in Linux kernel 2.6.x. Hopefully somebody could fix the related kernel code, so top can correctly show the number. Meanwhile, you can check the size of dirty pages by checking /proc/<pid>/smaps. The statistics are broken down within each VMA (Virtual Memory Area) region.

A dirty page means that a file-backed page contains modified data. This occurs when the kernel want to write to the disk. Except in direct I/O cases, the data are just written to page cache and the page itself is marked as dirty. Later, pdflush kernel threads periodically scan those dirty pages and write their contents to the disk. This asynchronous procedure prevents the writing task from being blocked too long.

Now, we turn our attention to NI and PR field. These fields represents priority of a process. NI displays the nice level, a static priority assigned to a task when it is initialized. By default, every new process has nice level 0, but you can override it with the "nice" utility. NI ranges from -20 to 19. PR shows the dynamic priority, which is calculated based on the nice level. When a process begins its life, PR equals NI plus 20.

During runtime, PR might go up or down depending on the kernel version you use. Pre-2.6.23 Linux kernel, PR can be any value within NI+20-x and NI+20+x. x itself could be considered as "bonus" or "discount" point. If a process sleeps a lot, it will get bonus so its PR is decremented. Otherwise, if a process chews a lot of CPU time, its PR will be incremented. When the kernel scheduler wants to choose which process should run, the process with the lowest dynamic priority wins.

In Linux 2.6.23 and above, PR is always NI plus 20. It is the consequence of the newly merged CFS scheduler, where sleep interval no longer solely dictates dynamic priority adjustment. And, now dynamic priority is not the main concern when selecting the next running task. Without going too deep into CFS internals, it is sufficient to say that the CFS scheduler selects the runnable process that currently consumes the least CPU virtual time. This is the "fairness" CFS wants to achieve, with every process having a proportional amount of CPU processing power, with PR acting as weighing factor to this fairness. Then we can conclude, with the implementation of CFS, a process with higher PR has a chance to preempt the lower one.

Know what is happening in “real time” on your systems is in my opinion the basis to use and optimize your OS. On ArchLinux or better on GNU/Linux in general the top command can help us, this is a very useful system monitor that is really easy to use, and that can also allows us to understand why our OS suffers and which process use most resources. The command to be run on the terminal is:

$ top

And we’ll get a screen similar to the one on the right:

Let’s see now every single row of this output to explain all the information found within the screen.

1° Row — top

topr1

This first line indicates in order:

current time (11:37:19)

uptime of the machine (up 1 day, 1:25)

users sessions logged in (3 users)

average load on the system (load average: 0.02, 0.12, 0.07) the 3 values refer to the last minute, five minutes and 15 minutes.

2° Row – task

topr2

The second row gives the following information:

Processes running in totals (73 total)

Processes running (2 running)

Processes sleeping (71 sleeping)

Processes stopped (0 stopped)

Processes waiting to be stoppati from the parent process (0 zombie)

3° Row – cpu

topr3

The third line indicates how the cpu is used. If you sum up all the percentages the total will be 100% of the cpu. Let’s see what these values indicate in order:

Percentage of the CPU for user processes (0.3%us)

Percentage of the CPU for system processes (0.0%sy)

Percentage of the CPU processes with priority upgrade nice (0.0%ni)

Percentage of the CPU not used (99,4%id)

Percentage of the CPU processes waiting for I/O operations(0.0%wa)

Percentage of the CPU serving hardware interrupts (0.3% hi — Hardware IRQ

Percentage of the CPU serving software interrupts (0.0% si — Software Interrupts

The amount of CPU ‘stolen’ from this virtual machine by the hypervisor for other tasks (such as running another virtual machine) this will be 0 on desktop and server without Virtual machine. (0.0%st — Steal Time)

4° and 5° Rows – memory usage

topr45

The fourth and fifth rows respectively indicate the use of physical memory (RAM) and swap. In this order: Total memory in use, free, buffers cached. On this topic you can also read the following article

Following Rows — Processes list

topr6

And as last thing ordered by CPU usage (as default) there are the processes currently in use. Let’s see what information we can get in the different columns:

PID – l’ID of the process(4522)

USER – The user that is the owner of the process (root)

PR – priority of the process (15)

NI – The “NICE” value of the process (0)

VIRT – virtual memory used by the process (132m)

RES – physical memory used from the process (14m)

SHR – shared memory of the process (3204)

S – indicates the status of the process: S=sleep R=running Z=zombie (S)

%CPU – This is the percentage of CPU used by this process (0.3)

%MEM – This is the percentage of RAM used by the process (0.7)

TIME+ –This is the total time of activity of this process (0:17.75)

COMMAND – And this is the name of the process (bb\_monitor.pl)

Conclusions

Now that we have seen in detail all the information that the command “top” returns, it will be easier to understand the reason of excessive load and/or the slowing of the system.

A good alternative to “ TOP ” is “ HTOP “, an evolution of top with features really amazing.

Free command examples

Example 1: Display RAM details in Linux machine

free

Output:

total used free shared buffers cached

Mem: 8027952 4377300 3650652 0 103648 1630364

-/+ buffers/cache: 2643288 5384664

Swap: 15624188 608948 15015240

Let us see what this table for you.

Line 1: Indicates Memory details like total available RAM, used RAM, Shared RAM, RAM used for buffers, RAM used of caching content.

Line 2: Indicates total buffers/Cache used and free.

Line 3: Indicates total swap memory available, used swap and free swap memory size available.

Let us dig more in to these lines to better understand it as Linux user prospective.

Line 1:Mem: 8027952 4377300 3650652 0 103648 1630364

8027952 : Indicates memory/physical RAM available for your machine. These numbers are in KB's

4377300 : Indicates memory/RAM used by system. This include even buffers and cached data size as well.

3650652 : Indicates Total RAM free and available for new process to run.

0 : Indicates shared memory. This column is obsolete and may be removed in future releases of free.

103648 : Indicates total RAM buffered by different applications in Linux

1630364 : Indicates total RAM used for Caching of data for future purpose.

Puzzled with buffers and cache?

What is the difference between buffers and Cache?

A buffer is a temporary location to store data for a particular application and this data is not used by any other application. This is similar to bandwidth concept. When you try to send burst of data through network, if your network card is capable of sending less data, it will keep these huge amounts of data in buffer so that it can send data constantly in lesser speeds. In other hand Cache is a memory location to store frequently used data for faster access. Other difference between a buffer and a cache is that cache can be used multiple times where as buffer is used single time. And both are temporary store for your data processing.

Line 2: -/+ buffers/cache: 2643288 5384664

2643288 : This is actual size of used RAM which we get from RAM used -(buffers + cache)

A bit of mathematical calculation

Used RAM = +4377300

Used Buffers = -103648

Used Cache = -1630364

Actual Total used RAM is 4377300 -(103648+1630364)= 2643288

Then why my Linux machine is showing 4377300 as used RAM. This is because Linux counts cached RAM, Buffered RAM to this used RAM. But in future if any application want to use these buffers/cache, Linux will free it for you. To know more about this, visit this site.

5384664 : Indicates actual total RAM available, we get to this number by subtracting actual RAM used from total RAM available in the system.

Total RAM = +8027952

actual used RAM = -2643288

Total actual available RAM = 5384664

So from today on words don't complain that Linux ate your RAM, its our understanding of free command output which is the culprit and the teacher who thought us Linux. If any one asks what is the free RAM available, we have to give this number(5384664) instead of first line number(4377300) for free RAM available in your machine.

Line 3: Swap: 15624188 608948 15015240

This line indicates swap details like total SWAP size, used as well as free SWAP.

Swap is a virtual memory created on HDD to increase RAM size virtually. To know more about swap click here for creating swap partition and here for swap file creation.

Linuxnix-free-e-book

Example2: Display RAM in human readable formats like in KB's, MB's, GB's, TB's

free -k

free -m

free -g

free –tera

Output:

root@linuxnix.com:/home/surendra# free -k

total used free shared buffers cached

Mem: 8027952 5323952 2704000 0 116876 1626940

-/+ buffers/cache: 3580136 4447816

Swap: 15624188 603792 15020396

root@linuxnix.com:/home/surendra# free -m

total used free shared buffers cached

Mem: 7839 5197 2642 0 114 1588

-/+ buffers/cache: 3495 4344

Swap: 15257 589 14668

root@linuxnix.com:/home/surendra# free -g

total used free shared buffers cached

Mem: 7 5 2 0 0 1

-/+ buffers/cache: 3 4

Swap: 14 0 14

root@linuxnix.com:/home/surendra# free –tera

total used free shared buffers cached

Mem: 0 0 0 0 0 0

-/+ buffers/cache: 0 0

Swap: 0 0 0

Note: If you observe with option –tera, the RAM size shows as 0, this is because my RAM is just 8GB and the output of free command will not support floating point numbers.

Example3: My actual RAM is 8GB but it is showing 7GB when using -g option, whats wrong with my RAM. This is because free command by default counts 1024 as power instead of 1000. To see difference use –si

With out –si

free -g

Output:

total used free shared buffers cached

Mem: 7 5 2 0 0 1

-/+ buffers/cache: 3 4

Swap: 14 0 14

with –si

free –si -g

Output:

total used free shared buffers cached

Mem: 8 5 2 0 0 1

-/+ buffers/cache: 3 4

Swap: 15 0 15

Example 4: Want to see combined size of both swap as well as RAM use -t option

free -t

Output:

total used free shared buffers cached

Mem: 8027952 5369012 2658940 0 117228 1634396

-/+ buffers/cache: 3617388 4410564

Swap: 15624188 603788 15020400

Total: 23652140 5972800 17679340

Example 5: I want to see continuous varying of RAM usage for every second, use -s option to mention number of seconds

free -s 1

Output:

total used free shared buffers cached

Mem: 8027952 5370220 2657732 0 117376 1635144

-/+ buffers/cache: 3617700 4410252

Swap: 15624188 603788 15020400 total used free shared buffers cached

Mem: 8027952 5367244 2660708 0 117392 1635272

-/+ buffers/cache: 3614580 4413372

Swap: 15624188 603788 15020400 total used free shared buffers cached

Mem: 8027952 5367556 2660396 0 117392 1635272

-/+ buffers/cache: 3614892 4413060

Swap: 15624188 603788 15020400 total used free shared buffers cached

Mem: 8027952 5367388 2660564 0 117392 1635272

-/+ buffers/cache: 3614724 4413228

Swap: 15624188 603788 15020400

## vmstat Basics

vmstat provides a number of values and will typically be called using two numerical parameters.

* Example: vmstat 1 5
  + 1 -> the values will be re-measured and reported every second
  + 5 -> the values will be reported five times and then the program will stop

The first line of the report will contain the average values since the last time the computer was rebooted. All other lines in the report will represent their respective current values. Vmstat does not need any special user rights. It can run as a normal user.

[user@fedora9 ~]$ vmstat 1 5

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

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

3 0 0 44712 110052 623096 0 0 30 28 217 888 13 3 83 1 0

0 0 0 44408 110052 623096 0 0 0 0 88 1446 31 4 65 0 0

0 0 0 44524 110052 623096 0 0 0 0 84 872 11 2 87 0 0

0 0 0 44516 110052 623096 0 0 0 0 149 1429 18 5 77 0 0

0 0 0 44524 110052 623096 0 0 0 0 60 431 14 1 85 0 0

[user@fedora9 ~]$

### Meaning of the individual Values

(Source man vmstat):

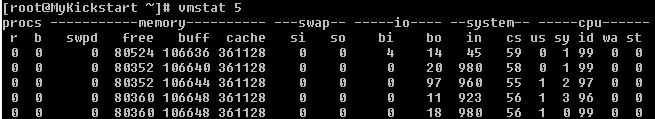
* Procs
  + r: The number of processes waiting for run time.
  + b: The number of processes in uninterruptible sleep.
* Memory
  + swpd: the amount of virtual memory used.
  + free: the amount of idle memory.
  + buff: the amount of memory used as buffers.
  + cache: the amount of memory used as cache.
  + inact: the amount of inactive memory. (-a option)
  + active: the amount of active memory. (-a option)
* Swap
  + si: Amount of memory swapped in from disk (/s).
  + so: Amount of memory swapped to disk (/s).
* IO
  + bi: Blocks received from a block device (blocks/s).
  + bo: Blocks sent to a block device (blocks/s).
* System
  + in: The number of interrupts per second, including the clock.
  + cs: The number of context switches per second.
* CPU
  + These are percentages of total CPU time.
  + us: Time spent running non-kernel code. (user time, including nice time)
  + sy: Time spent running kernel code. (system time)
  + id: Time spent idle. Prior to Linux 2.5.41, this includes IO-wait time.
  + wa: Time spent waiting for IO. Prior to Linux 2.5.41, included in idle.
  + st: Time stolen from a virtual machine. Prior to Linux 2.6.11, unknown.

# Performance tuning: vmstat output analysis

[Master](http://www.servercraftmen.com/author/anil/) July 14, 2013 [Home](http://www.servercraftmen.com/category/home/) [2 Comments](http://www.servercraftmen.com/vmstat-output-analysis/#comments)

By default vmstat and iostat generate a simple report showing the average values of different counters since the server has started. This information seems to be not much user friendly in most of the situations. But if you give the proper designated arguments for the command, it will generate an incremental report showing what  exactly the server is doing with your memory at the point of time. This will help us to fine tune your server for a better performance.

Lets have look at the vmstat output with an interval of 5 seconds.

[](http://servercraftmen.com/wp-content/uploads/2013/07/11.jpg)

The columns are grouped with the following headers.

1. procs  
2. memory  
3. swap  
4. io  
5. system  
6. cpu

**1. procs Header:**

The “procs” header is divided into 2 columns.

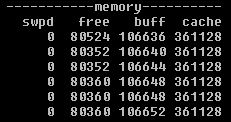
[2](http://servercraftmen.com/wp-content/uploads/2013/07/2.jpg)

**r column:** This column shows how many process are waiting for CPU time.

**b column:** This column shows the number of process in uninterrupted sleep, which means the process blocked on I/O [Disk I/O, Network I/O etc.]

**2. memory Header:**

Subdivided into 4 columns.

[](http://servercraftmen.com/wp-content/uploads/2013/07/3.jpg)

**swapd column:** This shows how many memory blocks are swapd out to disk. That means when the memory usage is full, few of the pages which are not recently accessed will be swapped to disk.

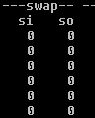
**free column:** The number of free memory blocks in the server

**buffers column:** The number of memory blocks used for buffers [1\*]

**cache column:** How many memory blocks used for operating cache [2\*]

**3. swap Header:**

This header shows the swap activity. Divided into two columns

[](http://servercraftmen.com/wp-content/uploads/2013/07/4.jpg)

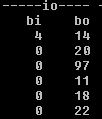
**si column:** This column shows how many memory blocks per second the operating system swapping in [from disk to memory]

**so column:** This column shows how many memory blocks per second the operating system swapping out [from memory to Disk]

This si and so columns are more important than the swapd column in the memory header group because this will give you a real time scenario about memory blocks swap in and swap out. If this value is more[say more than 10 or 15], the memory usage in the server can be concluded as critical.

**4. io Header:**

This column will give you the r/w per second on block devices.

[](http://servercraftmen.com/wp-content/uploads/2013/07/5.jpg)

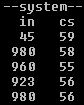
**bi column:** This column will tell you how many blocks per second read in from block devices.

**bo column:** This column will give you how many blocks per second wrote to block devices.

Usually this values will reflect the Disk I/O

**5. system Header:**

This header is sub-divided in to two columns.

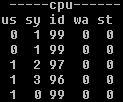
[](http://servercraftmen.com/wp-content/uploads/2013/07/6.jpg)

**in column:** This column will shows the number of interrupts per second. [3\*]

**cs column:** This will give you the number of context switches per second. [4\*]

**6. cpu Header:**

This header is sub-divided into 5 columns.

[](http://servercraftmen.com/wp-content/uploads/2013/07/7.jpg)

**us column:** This column shows the percentage of total CPU time spent for running the user code or user application [non-kernel codes]

**sy column:** This column shows the the percentage of total CPU time spent for running the system code/system process [kernel code]

**id column:** This shows the percentage idle time of the CPU.

**wa column:** This shows the percentage of CPU time waiting for I/O.

**st column:** This column shows the percent of CPU “stolen” from a virtual machine, if you are using virtualization. Please note that this column will not be there in the output if you are using a physical server. [5\*]

Now lets have some  more information about the terms used in the above [please have a look at the lines marked with <number>\*]

**1. Buffers [1\*]:** Buffers are a type of cache associated with the block devices and deals with the caching of file system metadata. When a process needs to access data from a file, the kernel will bring the data into main memory where the process will process it, and request that the data be saved in the file system back. When the kernel brings the data into memory, it also needs to bring the auxillary data associated with that data from the disk into memory. The auxiliary data will be any one of the following.

\* The super block of the file system that describes the free space available on the file system  
\* The inode which describes the layout of the file

The Caching of file system metadata is done in buffers and these buffers will remember the information about the file/directories [like permissions, contents etc.]

**2. Cache [2\*]:** Cache is also known as page cache. This cache only contains the contents of the files alone. When a file is read from disk or network, the contents are stored in cache. Also cache contains recently used memory pages, in case they’re needed again.

**3. Interrupts [3\*]:** An interrupt is a signal to the kernel that an event has occurred, and this results in changes in the sequence of instructions that is executed by the CPU. Two main types of interrupts, a hardware interrupt, is a signal to the system that has originated in hardware, such as the pressing of a key on the keyboard, a movement of the mouse or a progression in the system clock. Second one is software interrupts, generated by the application programs.

**4. Context Switches [4\*]:** A context switch is the switching of CPU from one process/thread to another. This is also known as process switch or task switch. So a context is the contents of a CPU’s registers and program counter at any point in time. So context switches enables multiple processes to share a single CPU and is an essential feature of a multitasking operating system.

**5. CPU steal time[5\*]:** This is the amount of real CPU time the VM hypervisor allocated to tasks other than running the virtual machine. ie: The steal time is the time CPU can be used for running the process, but VM hypervisor steal it for running something else instead of giving to CPU.

Vmstat is one of the key utilities for the System Administrators to troubleshoot as well as to optimize the performance of an existing system. Hope this post could give you an insight into “vmstat”. Stay tuned for more performance related articles !

lsof stands for List Open Files.

It is easy to remember lsof command if you think of it as “ls + of”, where ls stands for list, and of stands for open files.

It is a command line utility which is used to list the information about the files that are opened by various processes. In unix, everything is a file, ( pipes, sockets, directories, devices, etc.). So by using lsof, you can get the information about any opened files.

### 1. Introduction to lsof

Simply typing lsof will provide a list of all open files belonging to all active processes.

# lsof

COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME

init 1 root cwd DIR 8,1 4096 2 /

init 1 root txt REG 8,1 124704 917562 /sbin/init

init 1 root 0u CHR 1,3 0t0 4369 /dev/null

init 1 root 1u CHR 1,3 0t0 4369 /dev/null

init 1 root 2u CHR 1,3 0t0 4369 /dev/null

init 1 root 3r FIFO 0,8 0t0 6323 pipe

...

By default One file per line is displayed. Most of the columns are self explanatory. We will explain the details about couple of cryptic columns (FD and TYPE).

FD – Represents the file descriptor. Some of the values of FDs are,

* cwd – Current Working Directory
* txt – Text file
* mem – Memory mapped file
* mmap – Memory mapped device
* NUMBER – Represent the actual file descriptor. The character after the number i.e ’1u’, represents the mode in which the file is opened. r for read, w for write, u for read and write.

TYPE – Specifies the type of the file. Some of the values of TYPEs are,

* REG – Regular File
* DIR – Directory
* FIFO – First In First Out
* CHR – Character special file

For a complete list of FD & TYPE, refer man lsof.

# [Analyzing Database Server I/O Bottlenecks using IOSTAT](http://perumal.org/analyzing-database-server-io-bottlenecks-using-iostat/)

 Posted by [Ram Perumal](http://perumal.org/author/Ram/) at 5:30 am [Add comments](http://perumal.org/analyzing-database-server-io-bottlenecks-using-iostat/#respond)

May262009

iostat (input/output statistics) an utility that reports Central Processing Unit (CPU) statistics and input/output statistics for devices and partitions. The iostat command is used for monitoring system input/output device loading by observing the time the devices are active in relation to their average transfer rates. The iostat command generates reports that can be used to change system configuration to better balance the input/output load between physical disks.

The first report generated by the iostat command provides statistics concerning the time since the system was booted. Each subsequent report covers the time since the previous report. All statistics are reported each time the iostat command is run. The report consists of a CPU header row followed by a row of CPU statistics. On multiprocessor systems, CPU statistics are calculated system-wide as averages among all processors. A device header row is displayed followed by a line of statistics for each device that is configured.

Below is the sample default output, which displays a single history since boot report for all CPU and Devices.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10 | $ iostat  Linux 2.6.18-92.1.22.el5 (dblx131.lab.perumal.org)      05/26/2009    avg-cpu:  %user   %nice %system %iowait  %steal   %idle             1.24    0.88    0.55    0.83    0.00   96.51    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               2.89        34.98        52.43   19131771   28671500  dm-0              7.27        34.67        52.11   18960146   28500800  dm-1              0.08         0.31         0.31     168184     170696 |

The first report generated by the iostat command is the CPU Utilization Report. For multiprocessor systems, the CPU values are global averages among all processors. The report has the following format:

* %user: Show the percentage of CPU utilization that occurred while executing at the user level (application).
* %nice: Show the percentage of CPU utilization that occurred while executing at the user level with nice priority.
* %system: Show the percentage of CPU utilization that occurred while executing at the system level (kernel).
* %iowait: Show the percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.
* %steal: Show the percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor.
* %idle: Show the percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.

The second report generated by the iostat command is the Device Utilization Report. The device report provides statistics on a per physical device or partition basis. Block devices for which statistics are to be displayed may be entered on the command line. Partitions may also be entered on the command line providing that option -x is not used. If no device nor partition is entered, then statistics are displayed for every device used by the system, and providing that the kernel maintains statistics for it. If the ALL keyword is given on the command line, then statistics are displayed for every device defined by the system, including those that have never been used. The report may show the following fields, depending on the flags used:

* Device: This column gives the device (or partition) name.
* tps: Indicate the number of transfers per second that were issued to the device.
* Blk\_read/s: Indicate the amount of data read from the device expressed in a number of blocks per second.
* Blk\_wrtn/s: Indicate the amount of data written to the device expressed in a number of blocks per second.
* Blk\_read: The total number of blocks read.
* Blk\_wrtn: The total number of blocks written.
* kB\_read/s: Indicate the amount of data read from the device expressed in kilobytes per second.
* kB\_wrtn/s: Indicate the amount of data written to the device expressed in kilobytes per second.
* kB\_read: The total number of kilobytes read.
* kB\_wrtn: The total number of kilobytes written.
* MB\_read/s: Indicate the amount of data read from the device expressed in megabytes per second.
* MB\_wrtn/s: Indicate the amount of data written to the device expressed in megabytes per second.
* MB\_read: The total number of megabytes read.
* MB\_wrtn: The total number of megabytes written.
* rrqm/s: The number of read requests merged per second that were queued to the device.
* wrqm/s: The number of write requests merged per second that were queued to the device.
* r/s: The number of read requests that were issued to the device per second.
* w/s: The number of write requests that were issued to the device per second.
* rsec/s: The number of sectors read from the device per second.
* wsec/s: The number of sectors written to the device per second.
* rkB/s: The number of kilobytes read from the device per second.
* wkB/s: The number of kilobytes written to the device per second.
* rMB/s: The number of megabytes read from the device per second.
* wMB/s: The number of megabytes written to the device per second.
* avgrq-sz: The average size (in sectors) of the requests that were issued to the device.
* avgqu-sz: The average queue length of the requests that were issued to the device.
* await: The average time (in milliseconds) for I/O requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.
* svctm: The average service time (in milliseconds) for I/O requests that were issued to the device.
* %util: Percentage of CPU time during which I/O requests were issued to the device (bandwidth utilization for the device). Device saturation occurs when this value is close to 100%.

Below are some frequently used examples ;

* Display a continuous device report at two second intervals.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | $ iostat -d 2  Linux 2.6.18-92.1.22.el5 (dblx131.lab.perumal.org)      05/26/2009    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               2.90        34.99        52.45   19203049   28782692  dm-0              7.28        34.68        52.14   19031378   28611760  dm-1              0.08         0.31         0.31     168224     170928    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               2.00         0.00        48.00          0         96  dm-0              6.00         0.00        48.00          0         96  dm-1              0.00         0.00         0.00          0          0  ....  ....  .... |

* Display three reports at two second intervals for all devices.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | $ iostat -d 2 3  Linux 2.6.18-92.1.22.el5 (dblx131.lab.perumal.org)      05/26/2009    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               2.90        34.98        52.45   19203113   28789580  dm-0              7.28        34.67        52.14   19031442   28618648  dm-1              0.08         0.31         0.31     168224     170928    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               2.01         0.00        48.24          0         96  dm-0              6.03         0.00        48.24          0         96  dm-1              0.00         0.00         0.00          0          0    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               3.00         0.00        84.00          0        168  dm-0             10.50         0.00        84.00          0        168  dm-1              0.00         0.00         0.00          0          0 |

* Display three reports of extended statistics at two second intervals for devices hda and hdb.

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20 | $ iostat -x hda 2 3  Linux 2.6.18-92.1.22.el5 (dblx131.lab.perumal.org)      05/26/2009    avg-cpu:  %user   %nice %system %iowait  %steal   %idle             1.24    0.88    0.55    0.84    0.00   96.49    Device:         rrqm/s   wrqm/s   r/s   w/s   rsec/s   wsec/s avgrq-sz avgqu-sz   await  svctm  %util  hda               0.13     4.33  0.68  2.22    34.98    52.45    30.14     0.11   37.77   4.11   1.19    avg-cpu:  %user   %nice %system %iowait  %steal   %idle             6.47    0.00    1.49    0.50    0.00   91.54    Device:         rrqm/s   wrqm/s   r/s   w/s   rsec/s   wsec/s avgrq-sz avgqu-sz   await  svctm  %util  hda               0.00     3.98  0.50  3.48     3.98    59.70    16.00     0.01    1.88   1.50   0.60    avg-cpu:  %user   %nice %system %iowait  %steal   %idle             0.50    0.00    0.00    0.00    0.00   99.50    Device:         rrqm/s   wrqm/s   r/s   w/s   rsec/s   wsec/s avgrq-sz avgqu-sz   await  svctm  %util  hda               0.00     4.00  0.00  1.00     0.00    40.00    40.00     0.00    1.50   1.50   0.15 |

* Display three reports at two second intervals for device hda and all its partitions (hda1, etc.)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | $ iostat -p hda 2 3  Linux 2.6.18-92.1.22.el5 (dblx131.lab.perumal.org)      05/26/2009    avg-cpu:  %user   %nice %system %iowait  %steal   %idle             1.24    0.88    0.55    0.84    0.00   96.49    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               2.90        34.98        52.45   19203121   28793028  hda2              7.36        34.97        52.45   19200291   28793024  hda1              0.00         0.00         0.00       2302          4    avg-cpu:  %user   %nice %system %iowait  %steal   %idle             0.00    0.00    0.50    0.00    0.00   99.50    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               3.02         0.00        84.42          0        168  hda2             10.55         0.00        84.42          0        168  hda1              0.00         0.00         0.00          0          0    avg-cpu:  %user   %nice %system %iowait  %steal   %idle             0.00    0.00    0.00    0.00    0.00  100.00    Device:            tps   Blk\_read/s   Blk\_wrtn/s   Blk\_read   Blk\_wrtn  hda               0.00         0.00         0.00          0          0  hda2              0.00         0.00         0.00          0          0  hda1              0.00         0.00         0.00          0          0 |

Below are some general tips, which you can use while analyzing the output –

* Look for any devices with abnormal high blocks reads (Blk\_read/s) and writes (Blk\_wrtn/s) per second
* If %util is higher or near 100% utilization then it is a strong indicator for I/O bottleneck

If one of these disks are used by Oracle then you can check the data dictionary tables v$sql to see which SQL statement has generated most read/write and v$segment\_statistics to see which object has generated most read/write activity.

## What SAR can do?

In this article I will demonstrate how to install and configure sysstat package (which contains sar utility) and explains how to monitor the following Linux performance statistics using sar.

* 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

The statistics reported by sar deal with I/O transfer rates, paging activity, process-related activities, interrupts, network activity, memory and swap space utilization, CPU utilization, kernel activities and TTY statistics, among others. Both UP and SMP machines are fully supported.

# Examples of using SAR command for system monitoring in Linux

Submitted by [Sarath Pillai](http://plus.google.com/110882921997234815487?rel=author) on Mon, 11/26/2012 - 03:42



System Activity Reporter is an important tool that helps system administrators to get an overview of the server machine with status of different important metrics at different points of time.

If suppose you are having an issue with the system currently, Like some of your customers are unable to list some data from the database. The first thing that most of the Linux system administrators do is to recall the same issue when it previously occurred, and If you remember the day of its previous occurrence then you can easily compare the internal system statistics with the current statistics.

SAR is very much helpful in doing exactly that.

The first thing that we need to do is check and confirm whether you have SAR utility installed on the machine. Which can be checked by listing all rpm's and finding for this utility.

SAR is one of the utility inside sysstat. You can easily download and install it in your machine very easily through YUM. (But yeah dont worry because most of the distribution comes prepacked with sysstat toolsmiley).

[root@myvm1 ~]# yum install sysstat

Yeah but make it sure that you have epel,rpmformge repository enabled for installing. Otherwise your distribution DVD will be a nice place to look for the package.

## SAR (System Activity Reporter) will Give Information about the following things:

1. System Buffer activity
2. Information about system calls
3. Block device information
4. Overall paging information
5. Semaphore and memory allocation information
6. CPU utilization and process report

The main thing that we need to understand regarding SAR is that, everything is done using a cron. By default in many Linux distribution you will have a file named /etc/cron.d/sysstat.

**Lets see how really SAR works.**

If we start thinking about system monitoring, then the tool must have each and every data about the system's different aspects and must cover all time intervals. Which means a monitoring system must be able to provide the statistics of the machine for a given time.

There is no way, other than taking all the metrics and statistics of the machine at a definite time interval. Reducing the time interval for collecting the statistics will increase the amount of detailed statistics we have(because we will be having more data about the system).

SAR does exactly that. sar takes the statistics of different aspects of the machine at a definite time interval. So SAR runs through CRON.

[root@myvm ~]# cat /etc/cron.d/sysstat

# run system activity accounting tool every 10 minutes

\*/10 \* \* \* \* root /usr/lib64/sa/sa1 1 1

# generate a daily summary of process accounting at 23:53

53 23 \* \* \* root /usr/lib64/sa/sa2 -A

* So it can be seen from the above cron file for SAR that its running "sa1" script located at "/usr/lib64/sa/" at every 10 minutes
* And is also running a script /usr/lib64/sa/sa2 at the end of the day at around 23.53

So the first cron entry for SAR(/usr/lib64/sa/sa1) will run every 10 minutes which inturn will call the sadc utility to collect system stats and store it in a binary file (one file for a day)

And the second cron entry will dump all the contents of that binary file into another text file, and purges data older than a particular number of days, Normally 7 days by default(which is mentioned in the following file),

[root@archive ~]# cat /etc/sysconfig/sysstat

# How long to keep log files (days), maximum is a month

HISTORY=7

So you can modify that HISTORY entry easily by editing the file.

So although the system statistics is being collected every 10 minutes through cron(modify the cron to run every 1 minute for more accurate information) If you want to see the stats, then you need to run the command as below.

The simple sar command output is as shown below.

12:00:01 AM CPU %user %nice %system %iowait %steal %idle

12:01:01 AM all 73.28 0.00 1.25 0.00 0.00 25.47

12:02:01 AM all 7.83 0.00 0.44 0.00 0.00 91.73

12:03:01 AM all 61.65 0.00 0.70 0.00 0.00 37.66

12:04:01 AM all 57.85 0.00 0.82 0.00 0.00 41.34

12:05:01 AM all 4.25 0.00 0.41 0.00 0.00 95.34

12:06:01 AM all 4.20 0.00 0.22 0.00 0.00 95.58

12:07:01 AM all 5.05 0.00 0.33 0.00 0.00 94.63

12:08:01 AM all 4.76 0.00 0.06 0.00 0.00 95.18

12:09:01 AM all 37.57 0.00 0.37 0.00 0.00 62.05

12:10:01 AM all 70.04 0.00 0.80 0.00 0.00 29.16

12:11:01 AM all 5.03 0.00 0.12 0.00 0.00 94.84

It can be seen from the output that its reporting me the output of the collected stats for every minute(which means i have my cron at 1 minute interval), and will show the details of the whole day(or will show details collected till when you typed the command).

## Understanding the output of SAR command

**%user**: This shows the total time that the processor is spending on different process YCX5UKN5ZKEJ

**%sys:**this shows the percentage of time spend by the processor for operating system tasks(because the previous user shows the time spend for user end process)

**%iowait:**the name iowait itself suggests that its the time spend by processor waiting for devices(input and output)

**%nice:**Most of you guys must be knowing that a user can change the priority of a [process in linux](http://www.slashroot.in/linux-processes-administration-and-monitoring-tutorial)by [changing the nice value in Linux](http://slashroot.in/nice-and-renice-command-usage-examples-process-priority-linux). This table shows the time spend by CPU for process whose nice value has been changed.

**%steal:**This column shows the amount to time spend by a CPU (which is virtualized), for resources from the physical CPU

**%idle:**This suggests the idle time spend by the processor.

By default sar stores all of its data under /var/log/sa/ and a days are named as shown below.

s01 - for first day of the month

s02-for second day of the month

s03,s04..and so on.

## -d option in SAR command

This -d option can be used to report each and every activity related to different devices attached to the system(block devices). A typical output of the sar command with -d option is shown below.

12:00:01 AM DEV tps rd\_sec/s wr\_sec/s avgrq-sz avgqu-sz await svctm %util

12:01:01 AM dev3-0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:01:01 AM dev3-64 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:01:01 AM dev8-0 55.62 9.98 8317.87 149.72 13.68 245.96 2.61 14.52

12:01:01 AM dev8-16 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:02:01 AM dev3-0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:02:01 AM dev3-64 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:02:01 AM dev8-0 1.55 0.00 35.29 22.70 0.01 7.26 1.86 0.29

12:02:01 AM dev8-16 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:03:01 AM dev3-0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:03:01 AM dev3-64 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

**DEV:** this column names devices on the machine, according to major and minor numbers of a Linux block device. You can check this by doing an ls -l in /dev directory. as shown below.

brw-r----- 1 root disk 8, 0 Nov 16 16:29 sda

brw-r----- 1 root disk 8, 1 Nov 16 16:29 sda1

brw-r----- 1 root disk 8, 2 Nov 16 16:29 sda2

in the above shown "ls -l" outut for "sda", major number is "8",and minor number is "0"...So you can easily identify the disk mentioned in the sar command output.

**tps:**tps stands for transfer per second, so it shows the transfer per second to that particular device

**rd\_sec/s**: this shows you the total number of sectors on that device which is being read

**wr\_sec/s**: if rd\_sec/s is sectors being read per second then obviously wr\_sec is sectors being written per second.

**avgrq-sz**: this column shows the average.

**await:** this shows the total number of time that the processor waited for requests regarding IO

**%util:** this column shows the usage of cpu in percentage when the request was generated

## Show Memory usage in SAR command

the -r option available in sar command is very much useful. it shows the memory,swap,cached memory etc at every interval or required time interval.

02:20:01 AM kbmemfree kbmemused %memused kbbuffers kbcached kbswpfree kbswpused %swpused kbswpcad

02:30:01 AM 609500 1487652 70.94 242420 777560 1075980 364 0.03 360

02:40:01 AM 609500 1487652 70.94 242424 777568 1075980 364 0.03 360

02:50:01 AM 609500 1487652 70.94 242424 777592 1075980 364 0.03 360

03:00:01 AM 608980 1488172 70.96 242424 777600 1075980 364 0.03 360

03:10:01 AM 608584 1488568 70.98 242424 777628 1075980 364 0.03 360

03:20:01 AM 608584 1488568 70.98 242424 777648 1075980 364 0.03 360

in the above output most of the columns are self explanatory(and most of the outputs are in KB).

**kbmemfree:**this shows the amount of free memory

**Kbmemused:**memory used

**%memused:**percentage of memory used

**kbbuffers:**buffer memory used by the kernel.

**kbcached:**cached memory used by the kernel

all other entries for memory are swap(free,used,percentage etc)

## How to fetch metrics of a particular day using SAR in linux

As mentioned before all the metrics for a particular day are saved in sa<day of month> wise. So if i want to know my metrics for 27 th day the month i can easily find that out as shown below.

[root@archive ~]# sar -f /var/log/sa/sa27

Linux 2.6.18-194.el5xen (archive.r) 11/27/2012

02:20:01 AM CPU %user %nice %system %iowait %steal %idle

02:30:01 AM all 2.58 0.00 0.70 1.12 0.05 95.55

02:40:01 AM all 2.56 0.00 0.69 1.05 0.04 95.66

02:50:01 AM all 2.64 0.00 0.65 1.15 0.05 95.50

03:00:01 AM all 3.27 0.00 0.71 1.12 0.04 94.86

in the above command we have passed /var/log/sa/sa27 as an argument as i needed stats for that day..pass the sa<day of month>as you require in the above command

## How to fetch SAR metrics for a specific time on a particular date

this can be achieved by passing another argument as shown below.

[root@archive ~]# sar -f /var/log/sa/sa27 -s 02:20:00 -e 03:20:00

Linux 2.6.18-194.el5xen (archive.r) 11/27/2012

02:20:01 AM CPU %user %nice %system %iowait %steal %idle

02:30:01 AM all 2.58 0.00 0.70 1.12 0.05 95.55

02:40:01 AM all 2.56 0.00 0.69 1.05 0.04 95.66

02:50:01 AM all 2.64 0.00 0.65 1.15 0.05 95.50

03:00:01 AM all 3.27 0.00 0.71 1.12 0.04 94.86

03:10:01 AM all 2.72 1.06 0.75 1.09 0.04 94.33

Average: all 2.76 0.21 0.70 1.11 0.04 95.18

in the above shown example i asked sar to fetch the metrics between 2:20:00 and 3:20:00 on 27th day of the month

You can even pass any other metric option along with the time interval...such as -d or -r

Have you noticed ? SAR can accurately show us the machine statistics of a particular day at a particular time...so its much easier to identify the bottlenecks.

Using -A option along with the above command will show ALL (all the metrics collected by sar).

**sar -f /var/log/sa/sa27 -s 02:20:00 -e 03:20:00 -A**

The output will be elaborate. and you will get almost everything in sar from that -A option on your screen!

## Show network statistics using sar command

sar command even shows network statistics. This can be done by using the -n DEV option in sar command.

[root@archive ~]# sar -n DEV

Linux 2.6.18-194.el5xen (archive.r) 11/27/2012

02:20:01 AM IFACE rxpck/s txpck/s rxbyt/s txbyt/s rxcmp/s txcmp/s rxmcst/s

02:30:01 AM lo 0.01 0.01 0.77 0.77 0.00 0.00 0.00

02:30:01 AM eth0 12.30 0.12 1285.06 27.59 0.00 0.00 0.00

02:30:01 AM eth1 14.45 0.00 1399.34 0.00 0.00 0.00 0.00

02:30:01 AM sit0 0.00 0.00 0.00 0.00 0.00 0.00 0.00

02:40:01 AM lo 0.01 0.01 0.77 0.77 0.00 0.00 0.00

02:40:01 AM eth0 10.65 0.12 1139.38 27.00 0.00 0.00 0.00

02:40:01 AM eth1 13.96 0.00 1352.87 0.00 0.00 0.00 0.00

**IFACE:**stands for the nic card interface name

**rxpck/s:**this shows the total packets received per second

**txpck/s:**transmitted packets per second

**rxcmp/s**: compressed packets received

**txcmp/s**: compressed packets transmitted

**rxmcst/s:** packets multicasted per second.

## 

## Some other Metrics that can be determined using sar

**-y option in sar:**can be used to determine tty details

**-X option in sar** to get details of a particular process. You need to pass pid as an argument to this option.

**-n SOCK option in sar:**this option will report all socket details.

# Working with Standard Input, Output & Error in Linux

by [Pradeep Kumar](http://www.linuxtechi.com/author/pradeep/) · December 4, 2014

Every process in Linux is provided with three open files( usually called file descriptor). These files are the standard input, output and error files. By default :

* **Standard Input** is the keyboard, abstracted as a file to make it easier to write shell scripts.
* **Standard Output** is the shell window or the terminal from which the script runs, abstracted as a file to again make writing scripts & program easier
* **Standard error** is the same as standard output:the shell window or terminal from which the script runs.

A file descriptor is simply a number that refers to an open file. By default , file descriptor **0 (zero)** refers to the standard input & often abbreviated as **stdin**. File descriptor 1 refers to standard output (**stdout**) and file descriptor 2 refers to standard error (**stderr**). These numbers are important when you need to access a particular file , especially when you want to redirect these files to the other locations, File descriptors numbers go up from zero.

#### ****Redirecting Standard Output****

Syntax to redirect the output of a command to a file.

# Command\_options\_and\_arguments > output\_file

Example :

linuxtechi@localhost:~$ cat /proc/cpuinfo > command.txt

We can see the data that would have gone to the screen with more command :

linuxtechi@localhost:~$ more command.txt

processor     : 0

vendor\_id     : GenuineIntel

cpu family     : 6

model         : 37

model name     : Intel(R) Core(TM) i3 CPU       M 370  @ 2.40GHz

stepping     : 5

microcode     : 0x616

cpu MHz       : 0.000

cache size     : 6144 KB

physical id   : 0

siblings     : 2

core id       : 0

cpu cores     : 2

apicid         : 0

initial apicid    : 0

fpu         : yes

fpu\_exception     : yes

cpuid level     : 5

wp         : yes

The > operator tells the shell to redirect the output of the command to the given file. If the file exists , the deletes the old contents of the file and replaces it with the output of the command.

#### ****Redirecting a Command’s Input :****

Syntax to redirect the input of a command to come from a file.

# Command\_options\_and\_arguments < input\_file

Use the < operator to redirect the input for a command , example is shown below :

linuxtechi@localhost:~$ wc -l < command.txt

52

In this example , the input to the ‘**wc**‘ command comes from the file named command.txt. The shell sends the contents of the file command.txt as a standard input for the wc command.

**Note :** We can also combine both redirections with following syntax :

# command\_options\_and\_agruments < input\_file > output\_file.

#### ****Redirecting Standard Error :****

In addition to redirecting the standard input and output for a script or a command, we can also redirect standard error. Even though standard error by defaults goes to the same place as the standard output – the shell window or terminal. There are good reasons why stdout and stderr are treated separately. The main reason is that we can redirect the output of a command or commands to a file but you have no way of knowing whether an error occurred. Separating stderr from stdout allows the error message to appear on your screen while output still goes to a file.

Syntax to redirect stderr from a command to a file.

# command\_options\_and\_agruments 2> output\_file.

The 2 in **2>** refers to the file descriptor 2, the descriptor number for **stderr.**

Example:

linuxtechi@localhost:~$ lsash /usr/bin 2> commands-error.txt

linuxtechi@localhost:~$ cat commands-error.txt

No command 'lsash' found, did you mean:

Command 'sash' from package 'sash' (universe)

lsash: command not found

#### ****Redirecting both Standard Ouput & Standard Error.****

Use **2>&1** Syntax to redirect standard error to the same location as standard output .

Example:1

linuxtechi@localhost:~$ ls /usr/bin > command.txt 2>&1

Above Command has three parts.

* ls /usr/bin is the command run
* > command.txt redirects the output of the ls command
* 2>&1 sends the output of the file descriptor 2, stderr , to the same location as the file descriptor 1, stdout.

Example: 2

linuxtechi@localhost:~$ ls /usr2222/bin > command.txt 2>&1

linuxtechi@localhost:~$ more command.txt

ls: cannot access /usr2222/bin: No such file or directory

Note that above example assumes that your system doesn’t have directory names “/usr2222/bin”

#### ****Redirecting Both stderr & stdout at Once****

linuxtechi@localhost:~$ ls /usr2222/bin &> command.txt

linuxtechi@localhost:~$ more command.txt

ls: cannot access /usr2222/bin: No such file or directory

In the above command ls is the command , /usr2222/bin is the argument to the ‘**ls**‘ command and ‘**&> command.txt**‘ redirect both stdout and stderr to a file named command.txt.

Syslog is one of the most important standards used in Linux as it is the key file which helps you determine the different level of logs which are getting generated and stored every second while you are working on your Linux box. Syslog can be taken as "System Log".  
  
The main configuration file for syslog is  
  
For **RHEL 5** and older  
/etc/syslog.conf  
For **RHEL 6**  
/etc/rsyslog.conf

### Benefits of syslog

* Helps analyze the root cause for any trouble or problem caused
* Reduce overall downtime helping to troubleshoot issues faster with all the logs
* Improves incident management by active detection of issues
* Self-determination of incidents along with auto resolution
* Simplified architecture with different level of severity like error,info,[warning](javascript:void(0);) etc

The syslog.conf file is the main configuration file for the syslogd which logs system [messages](javascript:void(0);) on \*nix Systems.  This file specifies rules for logging. Every rule consists of two fields, a selector field and an action field.  These two fields are separated by one or more spaces or tabs.  The selector field specifies a pattern of facilities and priorities belonging to the specified action.

### Selectors

The selector field itself again consists of two parts, a facility and a priority, separated by a period (''.'').  Both parts are case insensitive.  
  
**For example**  
Kern.none, mail.info etc  
Here  
Kern = Facility  
None = severity or priority

### Facility

The facility is one of the following keywords: *auth*, *authpriv*, *cron*, [*daemon*](javascript:void(0);), *kern*, *lpr*, *mail*, *mark*, *news*,*security* (same as *auth*), *syslog*, *user*,*uucp* and *local0* through *local7*.  The keyword security should not be used anymore and mark is only for internal use and therefore should not be used in applications.  
  
Anyway, you may want to specify and redirect these messages here.  The facility specifies the subsystem that produced the message, i.e. all mail programs log with the mail facility (LOG\_MAIL) if they log using syslog.

### Facility Number

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Facility** | **Description** |
| **0** | kern | kernel messages |
| **1** | user | user level messages |
| **2** | mail | mail system |
| **3** | daemon | system daemons |
| **4** | auth | security/authorization messages |
| **5** | syslog | messages generated internally by syslogd |
| **6** | lpr | line printer subsystem |
| **7** | news | network news subsystem |
| **8** | uucp | UUCP subsystem |
| **9** | clock daemon |  |
| **10** | authpriv | security/authorization messages |
| **11** | ftp | FTP daemon |
| **12** | - | NTP susbsystem |
| **13** | - | log audit |
| **14** | - | log alert |
| **15** | cron | clock daemon |
| **16** | local0 | local use 0 (local0) |
| **17** | local1 | local use 1 (local1) |
| **18** | local2 | local use 2 (local2) |
| **19** | local3 | local use 3 (local3) |
| **20** | local4 | local use 4 (local4) |
| **21** | local5 | local use 5 (local5) |
| **22** | local6 | local use 6 (local6) |
| **23** | local7 | local use 7 (local7) |

### Severity Levels

The priority is one of the following keywords, in ascending order: *debug, info, notice, warning, warn (same as warning), err, error (same as err), crit, alert, emerg, panic (same as emerg)*.  The keywords error, warn and panic are deprecated and should not be used anymore. The priority defines the severity of the message

|  |  |
| --- | --- |
| **Integer** | **Facility** |
| **0** | Emergency: System is unusable |
| **1** | Alert: Action must be taken immediately |
| **2** | Critical: critical conditions |
| **3** | Error: Error conditions |
| **4** | Warning: Warning conditions |
| **5** | Notice: Normal but significant conditions |
| **6** | Informational: Informational messages |
| **7** | Debug: Debug level messages |

You can specify multiple facilities with the same priority pattern in one statement using the comma ('','') operator.  You may specify as much facilities as you want. Multiple selectors may be specified for a single action using the semicolon ('';'') separator.  Remember that each selector in the selector field is capable to overwrite the preceding ones.  Using this behavior you can exclude some priorities from the pattern.  
  
**Examples**  
 ***Log all the critical events on your Linux machine in a separate log file inside /var/log with a name of critical.log***  
Append this line inside /etc/syslog.conf  
\*.=crit            /var/log/critical.log  
***Log all the kernel related messages in separate log file inside /var/log/***[***firewall***](javascript:void(0);)***.log***  
Add a new line  
Kern.\*       /var/log/firewall.log  
  
Add a new entry at the end of the below line  
# Log anything (except mail) of level info or higher.  
# don’t log private authentication messages!  
# don’t log kernel related events and messages  
\*.info;mail.none;authpriv.none;cron.none;kern.none                /var/log/messages  
***Redirect all the error logs to a remote user root and Deepak on their terminals***  
# Messages of the priority alert will be directed  
# to the operator  
#  
\*.err                      root,deepak  
***Log all the firewall warning level messages inside /var/log/firewall-warning.log***  
Kern.warn                                           /var/log/firewall-warning.log

### Support for Remote Logging

These modifications provide [network support](javascript:void(0);) to the syslogd facility.  Network support means that messages can be forwarded from one node running syslogd to another node running syslogd where they will be actually logged to a disk file.  
  
The strategy is to have syslogd listen on a unix domain socket for locally generated log messages.  This behavior will allow syslogd to inter-operate with the syslog found in the standard C library.  At the same time syslogd listens on the standard syslog port for messages forwarded from other hosts.  To have this work  correctly the /etc/services file must have the following entry:   
Syslog 514/[udp](javascript:void(0);)  
If this entry is missing syslogd neither can receive remote messages nor send them, because the UDP port can’t be opened.  Instead syslogd will die immediately, blowing out an error message.  
  
**For example**,  
***to forward ALL messages to a***[***remote host***](javascript:void(0);)***uses the following syslog.conf entry:***  
                   # Sample syslogd configuration files to  
                   # Messages to a remote host forward all.  
                   \*.\*            @hostname  
***To forward all kernel messages to a remote host the configuration file would be as follows:***  
                   # Sample configuration files to forward all kernels  
                   # messages to a remote host.  
  
                   kern.\*         @hostname

How to give execute permission to all .php file under /tmp and subdirectory.

Chmod –R ugo+x /tmp/\*.php

total used free shared buffers cached

**Mem: 8027952 4377300 3650652 0 103648 1630364**

**-/+ buffers/cache: 2643288 5384664**

**Swap: 15624188 608948 15015240**

Let us see what this table for you.

**Line 1:** Indicates Memory details like total available RAM, used RAM, Shared RAM, RAM used for buffers, RAM used of caching content.  
**Line 2**: Indicates total buffers/Cache used and free.  
**Line 3:** Indicates total swap memory available, used swap and free swap memory size available.

Let us dig more in to these lines to better understand it as Linux user prospective.

**Line 1:Mem: 8027952 4377300 3650652 0 103648 1630364**

**8027952** : Indicates memory/physical RAM available for your machine. These numbers are in KB’s  
**4377300** : Indicates memory/RAM used by system. This include even buffers and cached data size as well.  
**3650652** : Indicates Total RAM free and available for new process to run.  
**0**:  Indicates shared memory. This column is obsolete and may be removed in future releases of free.  
**103648**: Indicates total RAM buffered by different applications in Linux  
**1630364** : Indicates total RAM used for Caching of data for future purpose.

**Puzzled with buffers and cache?**

# What is the difference between buffers and Cache?

A buffer is a temporary location to store data for a particular application and this data is not used by any other application. This is similar to bandwidth concept. When you try to send burst of data through network, if your network card is capable of sending less data, it will keep these huge amounts of data in buffer so that it can send data constantly in lesser speeds. In other hand Cache is a memory location to store frequently used data for faster access. Other difference between a buffer and a cache is that cache can be used multiple times where as buffer is used single time. And both are temporary store for your data processing.

**Line 2: -/+ buffers/cache: 2643288 5384664**

**2643288** : This is actual size of used RAM which we get from RAM used -(buffers + cache)

A bit of mathematical calculation

**Used RAM = +4377300**

**Used Buffers = -103648**

**Used Cache = -1630364**

**Actual Total used RAM is 4377300 -(103648+1630364)= 2643288**

Then why my Linux machine is showing 4377300 as used RAM. This is because Linux counts cached RAM, Buffered RAM to this used RAM. But in future if any application want to use these buffers/cache, Linux will free it for you. To know more about this, [**visit this site**](http://www.linuxatemyram.com/).

**5384664** : Indicates actual total RAM available, we get to this number by subtracting actual RAM used from total RAM available in the system.

**Total RAM = +8027952**

**actual used RAM = -2643288**

**Total actual available RAM = 5384664**

So from today on words don’t complain that Linux ate your RAM, its our understanding of free command output which is the culprit and the teacher who thought us Linux. If any one asks what is the free RAM available, we have to give this number(**5384664**) instead of first line number(**4377300**) for free RAM available in your machine.

**Line 3: Swap: 15624188 608948 15015240**

This line indicates swap details like total SWAP size, used as well as free SWAP.

Swap is a virtual memory created on HDD to increase RAM size virtually. To know more about swap[**click here**](http://www.linuxnix.com/2008/10/swap-management.html) for creating swap partition and [here for swap file creation](http://www.linuxnix.com/2008/10/linux-advanced-swap-management-swap-creation-virtual-file-system.html).

**Example2:** Display RAM in human readable formats like in KB’s, MB’s, GB’s, TB’s

**free -k**

**free -m**

**free -g**

**free --tera**

Output:

root@linuxnix.com:/home/surendra# free -k

total used free shared buffers cached

Mem: 8027952 5323952 2704000 0 116876 1626940

-/+ buffers/cache: 3580136 4447816

Swap: 15624188 603792 15020396

root@linuxnix.com:/home/surendra# free -m

total used free shared buffers cached

Mem: 7839 5197 2642 0 114 1588

-/+ buffers/cache: 3495 4344

Swap: 15257 589 14668

root@linuxnix.com:/home/surendra# free -g

total used free shared buffers cached

Mem: 7 5 2 0 0 1

-/+ buffers/cache: 3 4

Swap: 14 0 14

root@linuxnix.com:/home/surendra# free --tera

total used free shared buffers cached

Mem: 0 0 0 0 0 0

-/+ buffers/cache: 0 0

Swap: 0 0 0

**Note:**If you observe with option –tera, the RAM size shows as 0, this is because my RAM is just 8GB and the output of free command will not support floating point numbers.

**Example3:** My actual RAM is 8GB but it is showing 7GB when using -g option, whats wrong with my RAM. This is because free command by default counts 1024 as power instead of 1000. To see difference use –si

With out –si

**free -g**

Output:

total used free shared buffers cached

Mem: 7 5 2 0 0 1

-/+ buffers/cache: 3 4

Swap: 14 0 14

with –si

**free --si -g**

Output:

total used free shared buffers cached

Mem: 8 5 2 0 0 1

-/+ buffers/cache: 3 4

Swap: 15 0 15

**Example 4**: Want to see combined size of both swap as well as RAM use -t option

**free -t**

Output:

total used free shared buffers cached

Mem: 8027952 5369012 2658940 0 117228 1634396

-/+ buffers/cache: 3617388 4410564

Swap: 15624188 603788 15020400

Total: 23652140 5972800 17679340

**Example 5**: I want to see continuous varying of RAM usage for every second, use -s option to mention number of seconds

**free -s 1**

Output:

total used free shared buffers cached

Mem: 8027952 5370220 2657732 0 117376 1635144

-/+ buffers/cache: 3617700 4410252

Swap: 15624188 603788 15020400 total used free shared buffers cached

Mem: 8027952 5367244 2660708 0 117392 1635272

-/+ buffers/cache: 3614580 4413372

Swap: 15624188 603788 15020400 total used free shared buffers cached

Mem: 8027952 5367556 2660396 0 117392 1635272

-/+ buffers/cache: 3614892 4413060

Swap: 15624188 603788 15020400 total used free shared buffers cached

Mem: 8027952 5367388 2660564 0 117392 1635272

-/+ buffers/cache: 3614724 4413228

Swap: 15624188 603788 15020400

This is not a good way to display continuous output, we can use[**watch command**](http://www.linuxnix.com/2013/05/linux-watch-command-to-monitor-a-command.html)to do iterations

**watch free**

In out next post we will see other RAM details.