Linux Server Monitoring Guide

1. Real-time Monitoring with `top`

The classic system monitor that shows real-time process information.

```bash
# Basic top command
top

# Sort by CPU usage (default)
top -o %CPU

# Sort by memory usage
top -o %MEM

# Show specific user processes only
top -u username
```

Key Features:

- Real-time updates
- Shows system load, memory usage, and running processes
- Interactive - press 'q' to quit, 'h' for help
- Press 'P' to sort by CPU, 'M' to sort by memory

---

2. Better Alternative: `htop`

Enhanced version of top with better visualization and user interface.

```bash
# Install htop if not available
sudo apt install htop

# Run htop
htop
```

htop Features:

- Colorful interface with progress bars
- Press F6 to sort by different columns
- Press 'u' to filter by user
- Press 't' to show tree view
- Mouse support for scrolling and selection

---

3. One-time Snapshot with `ps`

Process status command for detailed process information.

```bash
# Top 10 processes by CPU usage
ps aux --sort=-%cpu | head -11

# Top 10 processes by memory usage
ps aux --sort=-%mem | head -11

# Show processes by specific user
ps aux | grep username

# Show all processes with user info sorted by CPU
ps -eo pid,ppid,cmd,%mem,%cpu,user --sort=-%cpu | head -20
```

Explanation of ps options:

- `aux` = show all processes with detailed info
- `--sort=-%cpu` = sort by CPU usage (descending)
- `-eo` = specify custom output format
- `head -11` = show top 10 results (plus header)

---

4. Check Per-User Resource Usage

Aggregate resource usage statistics grouped by user.

```bash
```

```
# Show CPU and memory usage by user
ps -eo user,%cpu,%mem --sort=-%cpu | awk 'NR>1 {cpu[$1]+=$2; mem[$1]+=$3; count[$1]++} END {printf "%-10s %(

# Simpler version - memory usage by user in MB
ps hax -o rss,user | awk '{a[$2]+=$1;}END{for(i in a)print i" "int(a[i]/1024)"MB";}' | sort -rnk2
```

What this shows:

- Total CPU percentage per user
- Total memory usage per user
- Number of processes per user
- Results sorted by resource consumption

---

5. System Resource Overview

General system statistics and information.

```bash
# Overall system stats
free -h       # Memory usage in human-readable format
uptime        # System uptime and load average
iostat        # I/O statistics (requires sysstat package)
vmstat 1 5    # Virtual memory stats (1-second intervals, 5 times)

# Check logged in users
w             # Who is logged in and what they're doing
who           # Simple list of logged in users
```

Key Information:

- `free -h`: Total, used, free, and available memory
- `uptime`: Load averages for 1, 5, and 15 minutes
- `w`: Shows user login time, idle time, and current processes
- `vmstat`: Memory, swap, I/O, and CPU statistics

---

6. Advanced Monitoring with `iotop` and `nethogs`

Specialized tools for I/O and network monitoring.

```bash
# Install if needed
sudo apt install iotop nethogs

# Monitor disk I/O by process/user
sudo iotop -o

# Monitor network usage by process
sudo nethogs
```

iotop Features:

- Shows disk read/write activity per process
- `-o` flag shows only processes with I/O activity
- Real-time updates of disk usage

nethogs Features:

- Network traffic monitoring per process
- Shows bandwidth usage by application
- Useful for identifying network-heavy processes

---

7. Check System Load and Processes

Low-level system information from /proc filesystem.

```bash
# System load
cat /proc/loadavg

# Memory information
cat /proc/meminfo

# Running processes count
ps aux | wc -l

# Processes per user
ps hax -o user | sort | uniq -c | sort -rn
```

Understanding load average:

- Numbers represent 1, 5, and 15-minute load averages
- Value of 1.0 means system is fully utilized
- Values > 1.0 indicate system overload

---

8. Quick One-liners for Top Resource Users

Efficient commands to quickly identify resource-heavy users.

```bash
# Top CPU consuming users
ps -eo user,%cpu --no-headers | awk '{cpu[$1]+=$2} END{for(u in cpu) print cpu[u], u}' | sort -rn | head -5

# Top memory consuming users
ps -eo user,%mem --no-headers | awk '{mem[$1]+=$2} END{for(u in mem) print mem[u], u}' | sort -rn | head -5
```

How it works:

1. Extract user and resource usage data
2. Use awk to sum usage by user
3. Sort results in descending order
4. Show top 5 users

---

9. Continuous Monitoring

Commands for ongoing system observation.

```bash
# Watch top processes every 2 seconds
watch -n 2 'ps aux --sort=-%cpu | head -10'

# Monitor specific user continuously
watch -n 2 'ps -u username -o pid,ppid,cmd,%mem,%cpu --sort=-%cpu'
```

watch command options:

- `-n 2` = refresh every 2 seconds
- Continuously executes the specified command
- Press Ctrl+C to stop monitoring

10. Finding and Killing Processes

Essential commands for process management and termination.

```bash
# Find processes by name
pgrep process_name
ps aux | grep process_name
pidof process_name

# Find processes by user
ps -u username
pgrep -u username

# Find processes using specific port
lsof -i :8080
netstat -tulpn | grep :8080
ss -tulpn | grep :8080

# Find processes using specific file/directory
lsof /path/to/file
fuser -v /path/to/directory

# Kill processes by PID
kill PID
kill -9 PID       # Force kill
kill -15 PID      # Graceful termination (SIGTERM)

# Kill processes by name
killall process_name
pkill process_name
pkill -u username  # Kill all processes by user

# Kill all processes matching pattern
pkill -f "pattern"
killall -9 process_name  # Force kill all instances
```

11. System Resource Monitoring

Comprehensive system health monitoring commands.

bash

```
# CPU information
lscpu                # CPU architecture details
cat /proc/cpuinfo    # Detailed CPU information
nproc                # Number of processing units

# Memory monitoring
cat /proc/meminfo    # Detailed memory information
slabtop              # Kernel slab allocator info
smem -t              # Memory usage with totals

# Disk usage and monitoring
df -h                # Filesystem disk usage
du -sh /path/*       # Directory sizes
lsblk                # Block devices information
fdisk -l             # Disk partitions

# System temperature (if available)
sensors              # Hardware sensors
cat /sys/class/thermal/thermal_zone*/temp

# System uptime and load
uptime -p            # Pretty uptime format
cat /proc/loadavg    # Load averages
w -s                 # Short format who output
```

---

12. Network and Service Monitoring

Monitor network connections and system services.

bash

```bash
# Network connections
netstat -tulpn          # All listening ports with processes
ss -tulpn               # Modern replacement for netstat
lsof -i                 # All network connections

# Service management and monitoring
systemctl status service_name      # Check service status
systemctl list-units --type=service --state=running  # Running services
systemctl list-units --failed      # Failed services
service --status-all               # All services status (SysV)

# Active connections by service
ss -p | grep service_name
netstat -ap | grep service_name

# Monitor specific service resources
systemctl status service_name | grep -E "(Memory|CPU)"
```

---

13. Advanced Process Analysis

Deep dive into process behavior and resource consumption.

```bash
# Process tree view
pstree                  # Show process hierarchy
pstree -p               # Include PIDs
ps f                    # Forest view of processes

# Detailed process information
cat /proc/PID/status    # Detailed process status
cat /proc/PID/cmdline   # Command line that started process
ls -la /proc/PID/fd/    # File descriptors used by process

# Process resource limits
ulimit -a               # Current user limits
cat /proc/PID/limits    # Specific process limits

# Real-time process monitoring
strace -p PID           # System calls made by process
ltrace -p PID           # Library calls made by process
```

---

14. Automated Monitoring Scripts

Useful one-liners and scripts for automated monitoring.

bash

```bash
# Find top CPU consuming processes and kill them
ps aux --sort=-%cpu --no-headers | head -5 | awk '{print $2}' | xargs kill

# Monitor memory usage continuously
while true; do free -h; sleep 5; done

# Alert when memory usage exceeds threshold
free | awk 'NR==2{printf "Memory Usage: %s/%s (%.2f%%)\n", $3,$2,$3*100/$2; if($3*100/$2 > 80) print "WARNING: H

# Find and kill zombie processes
ps aux | awk '{if($8=="Z") print $2}' | xargs kill -9

# Monitor specific user processes
watch -n 1 'ps -u username --sort=-%cpu -o pid,ppid,cmd,%mem,%cpu'

# Log top processes to file
ps aux --sort=-%cpu | head -20 >> /var/log/top_processes.log

# Check for processes consuming more than X% CPU
ps aux | awk '$3 > 50 {print "High CPU:", $0}'

# Emergency kill all processes by user (be careful!)
pkill -u username
# or more aggressive:
pkill -9 -u username
```

---

15. System Performance Troubleshooting

Commands for diagnosing performance issues.

bash

```
# I/O performance
iostat -x 1 5          # Extended I/O stats
sar -u 1 5             # CPU utilization over time
sar -r 1 5             # Memory utilization over time

# Find processes causing high I/O wait
top -o %CPU | grep "wa"
ps aux --sort=-%cpu | head -10

# Check swap usage
swapon -s              # Swap usage summary
cat /proc/swaps        # Swap file information

# Find large files
find / -type f -size +100M 2>/dev/null | head -20
du -h --max-depth=1 / 2>/dev/null | sort -hr | head -20

# Check open file descriptors
lsof | wc -l           # Total open files
cat /proc/sys/fs/file-nr # File descriptor usage

# Process priority management
nice -n 10 command     # Start process with lower priority
renice -n 5 -p PID     # Change running process priority
```

Emergency Response Commands

Critical commands for system emergencies.

bash

```
# System emergency stops
shutdown -h now          # Immediate shutdown
reboot                   # Restart system
systemctl emergency      # Emergency mode

# Kill runaway processes
killall -9 process_name  # Force kill all instances
pkill -f pattern         # Kill by command pattern
kill -9 -1               # Kill all processes (dangerous!)

# Free up memory quickly
sync                     # Force write cached data to disk
echo 3 > /proc/sys/vm/drop_caches  # Clear filesystem cache
swapoff -a && swapon -a  # Clear swap

# Check system integrity
dmesg | tail -50         # Recent kernel messages
journalctl -xe           # Recent system logs
systemctl --failed       # Failed services
```

Quick Reference Summary

Command	Purpose	Usage
`htop`	Interactive system monitor	Best for real-time monitoring
`top`	Classic system monitor	Universal availability
`ps aux --sort=-%cpu`	CPU usage snapshot	Quick identification
`ps aux --sort=-%mem`	Memory usage snapshot	Memory troubleshooting
`kill -9 PID`	Force kill process	Emergency process termination
`killall process_name`	Kill all instances	Stop multiple processes
`pgrep process_name`	Find process by name	Process identification
`lsof -i :port`	Find process using port	Network troubleshooting
`free -h`	Memory overview	System memory status
`df -h`	Disk usage	Storage monitoring
`w`	User activity	See who's doing what
`iotop`	Disk I/O monitoring	I/O performance issues
`nethogs`	Network monitoring	Network usage analysis
`systemctl status`	Service status	Service monitoring
`watch`	Continuous monitoring	Automated observation

Best Practices

1. **Start with htop** - Most user-friendly for general monitoring
2. **Use ps for scripting** - Better for automated monitoring scripts
3. **Monitor regularly** - Establish baseline performance metrics
4. **Document findings** - Keep logs of performance issues
5. **Combine tools** - Use multiple commands for comprehensive analysis

---

Troubleshooting Tips

- **High CPU usage**: Check with `htop` or `top -o %CPU`
- **Memory issues**: Use `free -h` and `ps aux --sort=-%mem`
- **Slow I/O**: Monitor with `iotop -o`
- **Network problems**: Check with `nethogs`
- **User-specific issues**: Filter with `ps -u username`

This guide covers essential Linux server monitoring commands for system administrators and DevOps engineers. Save this reference for quick troubleshooting and performance monitoring.

By: SHAHJAHAN