# Exp 1 Unix General Purpose Utility Commands

## 🔹 Case Study 1: System Monitoring and User Interaction in a Shared Development Environment

### Scenario:

A team of developers shares a Linux server for code deployment and debugging. Each developer uses the terminal for various tasks including checking system status, writing scripts, and managing files.

### Command Usage Scenarios:

| Command | Use Case / Scenario |
|---|---|
| `echo` | Used in shell scripts or to display output, e.g., `echo "Build started at $(date)"`. |
| `clear` | Clears the cluttered terminal screen during debugging or long sessions. |
| `exit` | Used to log out from the terminal or end a shell script execution. |
| `date` | Displays current system date/time—useful for logging or confirming system time. |
| `time` | Measures how long a command or script takes to execute, e.g., `time ./build.sh`. |
| `uptime` | Checks how long the server has been running—useful for troubleshooting crashes. |
| `cal` | Checks the calendar, useful when scheduling scripts or meetings via terminal. |
| `cat` | Displays contents of configuration or log files quickly. |
| `tty` | Tells which terminal you're connected to, helpful in debugging user sessions. |
| `man` | Accesses the manual page for a command, e.g., `man grep` to see syntax and options. |
| `which` | Finds the location of installed utilities, e.g., `which python` shows Python binary path. |

| | |
|---|---|
| `history` | Views recently executed commands for repeating or debugging past commands. |
| `id` | Displays user ID, group ID, useful in permission troubleshooting. |
| `pwd` | Displays the present working directory; helps users verify their path. |
| `whoami` | Confirms logged-in user identity—especially important with multiple users. |

## ✅ Tasks to be Performed by Developers

| Task No. | Task Description | Command(s) Used |
|---|---|---|
| 1 | Display a custom message when a build or script starts | `echo "Build started at $(date)"` |
| 2 | Clean up the terminal before starting a new debugging session | `clear` |
| 3 | Safely exit a terminal session or shell script after completion | `exit` |
| 4 | Log the current system date and time before starting a process | `date` |
| 5 | Measure the performance or execution time of a build or script | `time ./build.sh` |
| 6 | Check how long the server has been up (for troubleshooting) | `uptime` |
| 7 | View the calendar to plan meeting times or script executions | `cal` |
| 8 | Read the contents of configuration files or logs | `cat config.txt` or `cat logs/error.log` |
| 9 | Identify which terminal device a user session is connected to | `tty` |

| | | |
|---|---|---|
| 10 | Learn how to use an unfamiliar command or check syntax | `man grep`, `man find` |
| 11 | Locate the path of installed programs or interpreters | `which python`, `which node` |
| 12 | Review past commands for reuse or debugging | `history` |
| 13 | Check your user ID and group for permission checks | `id` |
| 14 | Confirm your current working directory | `pwd` |
| 15 | Confirm the current logged-in user (especially in multi-user environments) | |

---

## 🔹 Case Study 2: System Administration - Networking, Printing & Mail Server Maintenance

### Scenario:

A system admin in a university lab environment is responsible for maintaining Linux machines used by students for assignments, printing reports, and sending lab notifications via email.

### Command Usage Scenarios:

| Command | Use Case / Scenario |
|---|---|
| `ping` | Checks network connectivity to an external server (e.g., `ping google.com`). |
| `ifconfig` | Views or configures network interfaces, especially during WiFi/Ethernet issues. |
| `pr` | Prepares formatted output for printing (e.g., `pr report.txt`). |
| `lp` / `lpr` | Sends files to the printer (e.g., `lp project_summary.pdf`). |
| `lpstat` | Checks the status of current print jobs and printer queues. |
| `lpq` | Displays the print queue in more detail. |

| | |
|---|---|
| `lprm` | Removes a specific print job from the queue. |
| `cancel` | Cancels a printing job entirely (e.g., `cancel 42`). |
| `mail` | Sends system-generated emails to users (e.g., notifying students of system maintenance). |

## Exp 2
## Execution of File System Management Commands like ls, cd, pwd, cat, mkdir, rmdir, rm, cp, mv, chmod, wc, piping and redirection, grep, tr, echo, sort, head, tail, diff,comm, less, more, file, type, wc, split, cmp, tar, find, vim, gzip, bzip2, unzip, locate,etc.

Case Study 2: File Management in a Software Development Team

Scenario:

A software development team is working on a Unix-based system to manage project files efficiently.

Implementation Steps:

1. Creating a Project Directory and Files:

mkdir ProjectX → Creates a new directory for the project.

cd ProjectX → Navigates to the project directory.

touch main.py README.md → Creates a Python file and a README.

2. Checking and Managing Files:

ls -l → Lists files with details.

cat README.md → Displays README contents.

nano README.md → Edits the README file.

3. Copying and Moving Files:

cp main.py backup.py → Creates a backup of the script.

mv backup.py old_version.py → Renames the backup file.

4. Archiving and Deleting Files:

rm old_version.py → Deletes the old backup file.

rm -r ProjectX → Deletes the entire project directory if needed.

5. Checking File Content:

head main.py → Displays the first 10 lines of the script.

tail -n 5 main.py → Shows the last 5 lines.

These case studies provide practical usage of user and file management commands in real-world scenarios. Let me know if you need further details!

## Exp 3 Execution of User Management Commands like who, whoami, su, sudo, login, logout, exit, passwd, useradd/adduser, usermod, userdel, groupadd, groupmod, groupdel, gpasswd, chown, chage, chgrp, chfn, etc.

Case Study 1: User Management in a Multi-User Unix System

Scenario:

A university has a shared Linux server for students and faculty to access programming resources. The system administrator needs to create user accounts, manage permissions, and ensure security.

Implementation Steps:

1. Creating Student and Faculty Accounts:

useradd student1 → Creates a student account.

useradd faculty1 → Creates a faculty account.

passwd student1 → Sets a password for student1.

2. Assigning Users to Groups:

groupadd students → Creates a group for students.

groupadd faculty → Creates a group for faculty.

usermod -aG students student1 → Adds student1 to the "students" group.

usermod -aG faculty faculty1 → Adds faculty1 to the "faculty" group.

3. Checking Logged-In Users:

who → Displays active users.

id student1 → Shows the UID and GID of student1.

4. Deleting a User Who Graduated:

userdel -r student1 → Removes student1 and their home directory.

5. Checking Password Expiration Policy:

chage -l faculty1 → Checks password expiry for faculty1.

**Exp 4 Execution of Process Management Commands like ps, pstree, nice, kill, pkill, killall, xkill, fg, bg, pgrep, renice, etc.**

## Scenario-Based Usage of Process Management Commands

| Command | Scenario | Usage |
|---|---|---|
| `ps` | You want to see all currently running processes for your user. | `ps aux` or `ps -ef` |
| `pstree` | You want to view the parent-child relationship of processes. | `pstree` |
| `nice` | You want to start a process with lower priority so it doesn't consume much CPU. | `nice -n 10 myscript.sh` |
| `renice` | You want to change the priority of an already running process. | `renice -n 5 -p <PID>` |
| `kill` | You want to terminate a process using its PID. | `kill 1234` |
| `pkill` | You want to kill a process by its name (not PID). | `pkill firefox` |
| `killall` | You want to kill all instances of a process. | `killall chrome` |
| `xkill` | A graphical app is frozen and you want to forcefully kill it by clicking on its window. | `xkill` (then click on the window) |
| `fg` | A process running in background needs to be brought to foreground. | `fg %1` |
| `bg` | A stopped process needs to be resumed in the background. | `bg %1` |
| `pgrep` | You want to find PID of a running process by name. | `pgrep python` |

## 📚 Case Study: Managing Application Performance in a Linux Server

**Problem:**

A software development company is running multiple apps (e.g., Python scripts, MySQL server, Firefox for testing) on a Linux server. Some scripts consume a lot of CPU, causing the system to slow down. The team needs to monitor and manage these processes efficiently.

**Objectives:**

- Identify high-CPU usage processes
- Reduce the CPU priority of some tasks
- Kill non-responsive processes
- Manage background/foreground jobs

---

## 🛠️ Solution Using Process Management Commands

### Step 1: Monitor Active Processes

bash
CopyEdit
```
ps aux --sort=-%cpu
```

🔹 *Lists all processes sorted by CPU usage.*

---

### Step 2: View Process Hierarchy

bash
CopyEdit
```
pstree -p
```

🔹 *Visualizes how child processes are linked to parent processes.*

---

### Step 3: Run Heavy Script with Lower Priority

bash
CopyEdit
```
nice -n 10 python3 heavy_script.py
```

🔹 *Prevents script from hogging CPU.*

---

### Step 4: Find the PID of a Resource-Hungry Process

bash
CopyEdit
```
pgrep -f heavy_script.py
```

🔹 *Returns the PID of the script if you didn't store it initially.*

---

### Step 5: Change the Priority of a Running Process

bash
CopyEdit
```
renice -n 15 -p 4567
```

🔹 *Reduces priority of PID 4567, giving more CPU to others.*

---

### Step 6: Kill a Hung Firefox Browser

bash
CopyEdit
```
pkill firefox
```

🔹 *Closes all Firefox processes.*

OR

bash
CopyEdit
```
xkill
```

🔹 *Click on a frozen window to forcefully close it.*

---

**Step 7: Background and Foreground Process Handling**

bash
CopyEdit
```
./long_task.sh &
jobs       # View job list
fg %1      # Bring job 1 to foreground
bg %1      # Send job 1 to background again
```

---

## ✅ Conclusion:

This case study shows how a Linux admin can use **process management commands** to:

- Optimize system performance,
- Reduce CPU load from non-critical tasks,
- Handle frozen or unresponsive applications,
- Maintain server health in a multi-user environment.

## Exp 5 Execution of Memory Management Commands like free, /proc/meminfo, top, htop,df, du, vmstat, demidecode, sar, pagesize, etc.

## Case Study: Diagnosing Memory Bottlenecks on a Production Server

**Background:**

A company runs a web server that has been responding slowly. The system administrator is assigned the task of diagnosing whether it's a **memory issue** or a **disk usage problem**.

---

## 🧠 SCENARIOS & COMMANDS USED

### 🟩 1. Monitor Available RAM in Real-Time

**Scenario:** You want to check whether your server is running out of memory.

**Command:**

bash
CopyEdit
```
free -h
```

- 
- **Use:** Displays free, used, and total memory (RAM and swap) in a human-readable format.
- **Solution:** If `free` shows low "available" RAM and high swap usage, it indicates memory pressure.

---

### 🟩 2. Check Detailed Memory Information

**Scenario:** You need exact stats on memory breakdown.

**Command:**

bash
CopyEdit
```
cat /proc/meminfo
```

- 
- **Use:** Dumps detailed memory stats directly from the kernel.
- **Solution:** Check `MemAvailable`, `Buffers`, `Cached` to understand actual memory usage.

---

## 🟩 3. Real-Time Process Monitoring

**Scenario:** Find out which processes consume the most memory.

**Command:**

bash
CopyEdit
```
top
```

- 
- **Use:** Lists running processes sorted by CPU/memory usage.
- **Solution:** Identify memory-hogging processes (look under `%MEM` column).

**Alternative Command:**

bash
CopyEdit
```
htop
```

- 
- **Use:** Colorful, interactive version of `top`.

- **Note:** May need to install it using `sudo apt install htop`.

---

## 🟩 4. Check Disk Space Usage

**Scenario:** Check if the system is running out of disk space, impacting swap and memory.

**Command:**

bash
CopyEdit
```
df -h
```

- 
- **Use:** Displays disk space usage for all mounted partitions.
- **Solution:** Identify if `/` or `/home` is nearly full.

---

## 🟩 5. Check Directory Size to Clear Space

**Scenario:** You need to identify which directories are taking up space.

**Command:**

bash
CopyEdit
```
du -sh /var/*
```

- 
- **Use:** Displays size of folders under `/var`.
- **Solution:** Delete large unnecessary logs from `/var/log`.

---

## 🟩 6. View Memory Statistics in Real-Time

**Scenario:** Need to observe memory usage pattern over time.

**Command:**

bash
CopyEdit
```
vmstat 5
```

- 
- **Use:** Displays memory, CPU, and I/O usage every 5 seconds.
- **Solution:** Helps identify spikes in memory demand or swap usage.

---

## 🟩 7. Check Hardware Details

**Scenario:** Want to verify total physical RAM and memory slots.

**Command:**

bash
CopyEdit
```
sudo dmidecode -t memory
```

- 
- **Use:** Provides hardware-level details of memory modules.
- **Solution:** Used to plan memory upgrades.

---

## 🟩 8. Check Page Size

**Scenario:** Need to know the system's memory page size (used in memory tuning).

**Command:**

bash

CopyEdit
```
getconf PAGE_SIZE
```

- 
- **Output:** Shows memory page size (e.g., 4096 bytes).
- **Use:** Useful in understanding paging behavior.

---

### 🟩 9. Historical Resource Usage Report

**Scenario:** Need to analyze memory usage over the last few hours/days.

**Command:**

bash
CopyEdit
```
sar -r 1 5
```

- 
- **Use:** Reports memory usage (free, used, cache, buffer) at 1-second intervals, 5 times.
- **Note:** You might need to install `sysstat` package.

---

## 🧪 Summary of the Case Study Execution

**Problem Observed:**

- Web server slow and unresponsive.

**Commands Used:**

- `free -h`, `top`, and `vmstat` revealed high memory usage.

- `cat /proc/meminfo` confirmed low available memory and swap usage.
- `df -h` showed root partition 90% full.
- `du -sh /var/*` identified large logs in `/var/log`.
- `htop` showed specific service using excessive RAM.
- `dmidecode` checked if more RAM could be added.

**Solution Taken:**

- Cleared unused logs.
- Restarted memory-intensive service.
- Scheduled system upgrade to add more RAM.

---

## ✅ Conclusion:

These commands together provide a **comprehensive diagnostic toolkit** for identifying, analyzing, and resolving memory-related issues on Unix/Linux systems.