

Project Report
On
User Management Project

MASTERS OF COMPUTER APPLICATIONS



Submitted By:
Komalpreet Kaur
(24MCA20195)

Project Guide:
Er. Prabhjot Kaur
MCA Dept., (CU)

DEPARTMENT OF COMPUTER APPLICATION,
(CHANDIGARH UNIVERSITY)
(NH05, Chandigarh-Ludhiana Highway , Gharuan, Mohali , Punjab , India)

SESSION 2024-26

DECLARATION

I, Komalpreet Kaur, hereby declare that this project report titled "*User Management*" is original work carried out by me under the supervision of Er. Prabhjot Kaur. I further declare that this work has not been submitted to any other institute/university for the award of the degree of Master of Computer Applications.

Student Name: **Komalpreet Kaur**

Roll No: **24MCA20195**

ACKNOWLEDGEMENT

I express my sincere gratitude to my project guide, Er. Prabhjot Kaur, for invaluable guidance and support throughout this project. I also extend my thanks to Chandigarh University for the opportunity to undertake this project and to my classmates and family for their continuous encouragement.

Komalpreet Kaur
24MCA20195

TABLE OF CONTENTS

| Chapter | Title | Page No. |
|---------|---------------------------------------|----------|
| 1 | Introduction | 6 |
| 1.1 | Scope of the system | 6 |
| 1.2 | Project Description | 6 |
| 1.2.1 | About Existing System | 6 |
| 1.2.2 | Implementation of Proposed System | 6 |
| 1.3 | Advantages of the project | 6 |
| 2 | Project Category Tools & Environment | 7 |
| 2.1 | Project Category | 7 |
| 2.2 | Front-end coverage | 7 |
| 2.3 | Back-end coverage | 7 |
| 2.4 | Software and Hardware requirements | 7 |
| 3 | Project Development Stages | 8 |
| 3.1 | Recognition of needs | 8 |
| 3.2 | Feasibility study | 8 |
| 3.3 | System Analysis (DFD, ER Diagrams) | 8 |
| 3.4 | Structure Charts, Data Tables | 8 |
| 3.5 | System Development | 8 |
| 3.6 | Testing, Implementation & Maintenance | 8 |
| 4 | Samples of Reports | 10-11 |
| 5 | Future Enhancement | 12-13 |
| 6 | Conclusion | 14 |
| 7 | Bibliography | 15 |

Chapter 1: Introduction

1.1 Scope of the System

1. Basic User and Group Management

- **Adding, Modifying, and Deleting Users:** Creating users with specific details (name, password, home directory, shell), modifying their information, and deleting users when no longer needed.
- **Group Management:** Creating, modifying, and deleting groups, as well as managing group memberships to control access.
- **Password Management and Security Policies** Password Policies: Enforcing password strength requirements, expiration, and rotation to improve system security.
- **Password Aging and Expiration:** Setting policies to require users to change passwords periodically and locking expired accounts.

1.2 Project Description

1.2.1 About Existing System

In a Linux user management project, the "**exit system**" (or "**exit management system**") refers to structured process of handling user accounts when they need to be disabled, deactivated, or removed—typically when a user leaves an organization, their role changes, or they no longer require access. This system is crucial for maintaining security, ensuring compliance, and protecting sensitive data.

1. User Account Locking

- **Purpose:** Immediately disables the user's ability to log in without fully removing the account or its data.

2. Session Termination

- **Purpose:** Ensures all active sessions and processes for the user are terminated to prevent any ongoing access.

1.2.2 Implementation of the Proposed System

Implementing a user management system in Linux involves creating a structured process for managing user accounts effectively and securely. This system should cover account creation, permission management, data access, and secure account termination. Below is a step-by-step guide to implementing a basic user management system, with a focus on key areas like account provisioning, monitoring, and exit management.

1. Define the User Management Policies

Purpose: Create policies that outline user roles, permissions, access controls, and exit protocols.

Documentation: Write a guide for administrators that includes steps for account setup, maintenance, security, and termination. This ensures consistent processes across all accounts.

2. Account Provisioning (User Creation)

- **Purpose:** Set up new user accounts according to defined roles and permissions.

1.3 Advantages of the Project

Implementing a user management system in Linux offers several benefits that enhance security, streamline administrative tasks, improve compliance, and optimize resource usage. Here's a look at the key advantages:

1. Enhanced Security

- **Controlled Access:** By managing users and permissions, you control who can access specific files, directories, and system resources, reducing the risk of unauthorized access.
- **Granular Permissions:** Groups and roles allow precise control over access to sensitive data and critical system operations. Users get access only to the resources they need for their role.

2. Streamlined Administration

- **Centralized User Management:** With clear policies and procedures, managing users becomes standardized, which reduces administrative overhead and simplifies user lifecycle management.
- **Automated Processes:** Routine tasks, such as creating users, assigning roles.

Chapter 2: Project Category, Tools & Environment

2.1 Project Category

System Administration Projects

System Administration Projects

These projects focus on tasks related to the configuration, management, and upkeep of systems, including user management.

- **User Account Creation and Management:** Setting up a system to create, modify, and delete user accounts based on role, security needs, and system requirements.
- **Permissions Management:** Assigning and enforcing permissions for files, directories, and resources based on user roles.
- **System Monitoring and Auditing:** Setting up tools like auditd or syslog to monitor user actions, login attempts, and other system behaviors

2.1 Back-end Coverage

2.2 Software and Hardware Requirements

Software:

OS: CentOS 7/8

Programming Language:

Python 3.x IDE: Visual Studio

Code / PyCharm

Libraries: PyQt5 (for future GUI implementation)

Hardware:

Processor: Intel i5 or

higher RAM: 4 GB

minimum

Disk Space: 50 GB minimum

Chapter 3: Project Development Stages

3.1 Recognition of Needs

Recognition of Needs is the critical first step in identifying the specific requirements, goals, and objectives of a project. It involves understanding the challenges, identifying gaps, and defining what is necessary to meet the desired outcomes. In the context of a Linux User Management Project, recognizing the needs would involve a thorough analysis of both technical and organizational requirements. Here's how you can approach this process:

3.2 Feasibility Study

A Feasibility Study is a crucial phase in the planning process of a project. It aims to assess the practicality and viability of the proposed system by evaluating various factors such as technical, financial, operational, and legal considerations. In the context of a Linux User Management Project, a feasibility study will help determine whether the project can be successfully executed and whether it is worth the investment of resources.

3.3 System Analysis (DFD, ER Diagrams)

Steps Involved in System Analysis for a Linux User Management Project

Requirement Gathering and Documentation

- a. **Identify Stakeholders:** Gather input from system administrators, end-users, security teams, compliance officers, and other relevant stakeholders. Their input will shape the system's requirements and features.

3.4 System Design (Structure Charts, Data Tables)

System design is the phase where the structure and architecture of the system are defined. It involves creating a detailed blueprint for the system, which specifies how all components of the system will interact, what data will be used, and how different processes will be executed. In the case of a Linux User Management Project, system design should address the creation, modification.

3.5 System Development

System development is the phase where the design specifications from the previous phases (analysis and design) are transformed into an actual working system. This phase involves programming, configuring, testing, and integrating the system's components based on the design documentation. For a Linux User Management System, this will include setting up user management functionalities, integrating security policies, ensuring user authentication and authorization.

3.6 Testing, Implementation & Maintenance

The system underwent functional testing to ensure proper execution of account creation, deposits, and withdrawals. It was implemented on CentOS and is maintained with updates to handle additional functionality and bug fixes.

Chapter 4: Sample Reports

In the current Linux environment, user and group management relies on a set of standard utilities and manual configuration of system files:

- **User and Group Management Commands:**

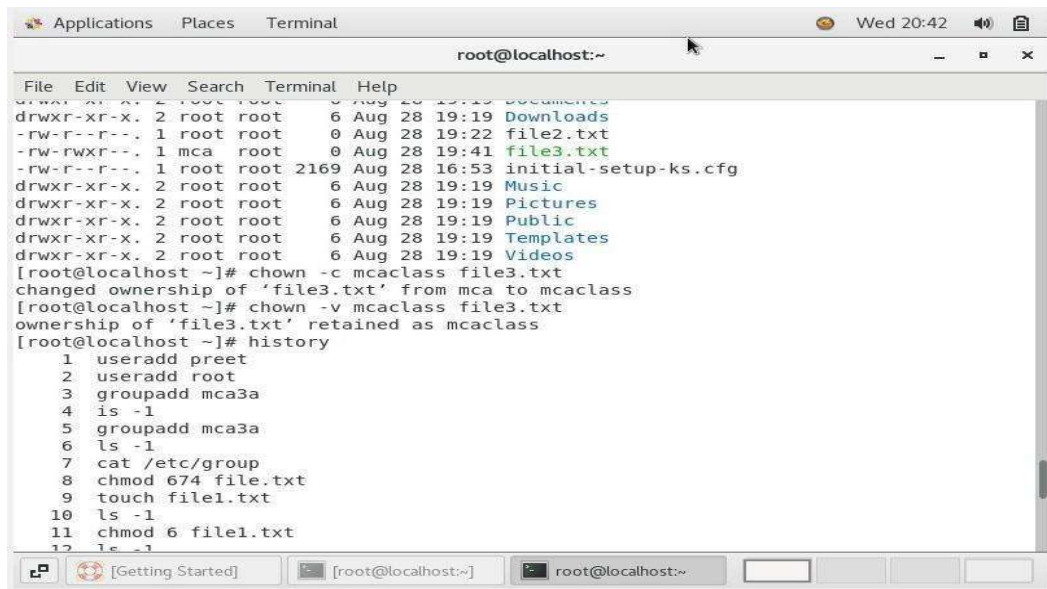
- `useradd`: Adds a new user to the system.
- `usermod`: Modifies an existing user's details (e.g., changing passwords, groups).
- `userdel`: Deletes a user and optionally their home directory.
- `groupadd`: Creates a new group.
- `groupdel`: Deletes a group.
- `passwd`: Changes a user's password.
- `chown`: Changes file ownership.
- `chmod`: Changes file permissions.

- **User Information Storage:**

- `/etc/passwd`: Contains user account information, including username, user ID (UID), group ID (GID), home directory, and shell.
- `/etc/shadow`: Stores user password hashes and password expiration information.
- `/etc/group`: Contains group information (group names, GIDs, and group members).
- `/var/log/auth.log` or `/var/log/secure`: Stores authentication logs, including successful and failed login attempts.
-

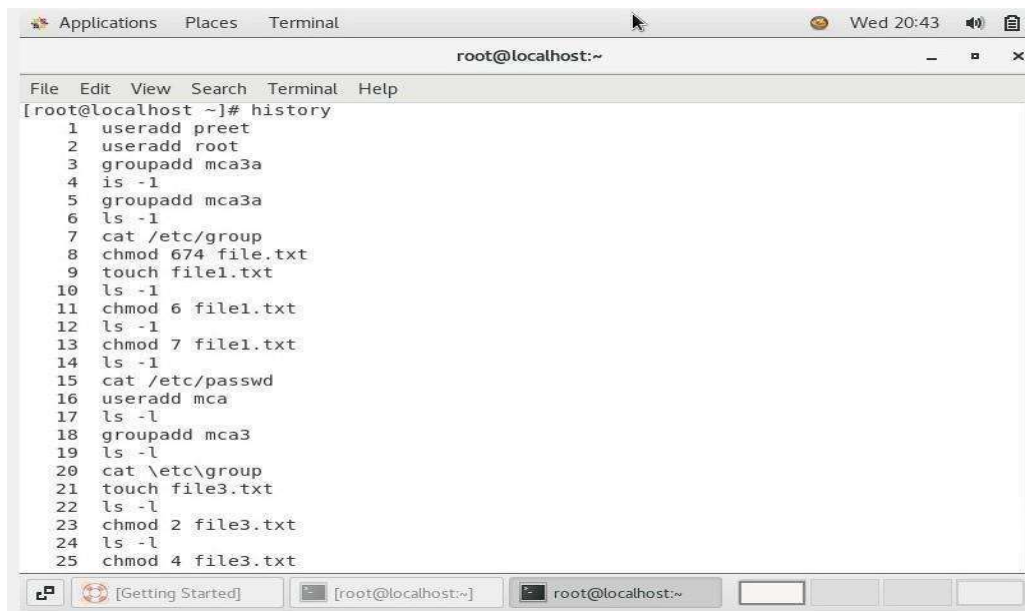
While these utilities are powerful, they require knowledge of command-line syntax and manual intervention for tasks such as bulk user creation, modification, and permission management. The existing system lacks a cohesive, automated tool for managing users and groups, which can be time-consuming and error-prone in large environments.

Output:



A terminal window titled 'root@localhost:~' showing the output of a series of commands. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal output shows the results of 'ls -l' for the home directory, followed by 'chown' and 'chown -v' commands to change ownership of 'file3.txt' from 'mca' to 'mcaclass'. Finally, the 'history' command is run, showing a list of 12 commands.

```
File Edit View Search Terminal Help
drwxr-xr-x. 2 root root  6 Aug 28 19:19 Documents
drwxr-xr-x. 2 root root  6 Aug 28 19:19 Downloads
-rw-r--r--. 1 root root  0 Aug 28 19:22 file2.txt
-rw-rw-r--. 1 mca  root  0 Aug 28 19:41 file3.txt
-rw-r--r--. 1 root root 2169 Aug 28 16:53 initial-setup-ks.cfg
drwxr-xr-x. 2 root root  6 Aug 28 19:19 Music
drwxr-xr-x. 2 root root  6 Aug 28 19:19 Pictures
drwxr-xr-x. 2 root root  6 Aug 28 19:19 Public
drwxr-xr-x. 2 root root  6 Aug 28 19:19 Templates
drwxr-xr-x. 2 root root  6 Aug 28 19:19 Videos
[root@localhost ~]# chown -c mcaclass file3.txt
changed ownership of 'file3.txt' from mca to mcaclass
[root@localhost ~]# chown -v mcaclass file3.txt
ownership of 'file3.txt' retained as mcaclass
[root@localhost ~]# history
 1  useradd preet
 2  useradd root
 3  groupadd mca3a
 4  ls -l
 5  groupadd mca3a
 6  ls -l
 7  cat /etc/group
 8  chmod 674 file.txt
 9  touch file1.txt
10  ls -l
11  chmod 6 file1.txt
12  ls -l
```



A terminal window titled 'root@localhost:~' showing the output of the 'history' command. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal output shows a list of 25 commands, including user and group management, file creation, and permission changes.

```
File Edit View Search Terminal Help
[root@localhost ~]# history
 1  useradd preet
 2  useradd root
 3  groupadd mca3a
 4  ls -l
 5  groupadd mca3a
 6  ls -l
 7  cat /etc/group
 8  chmod 674 file.txt
 9  touch file1.txt
10  ls -l
11  chmod 6 file1.txt
12  ls -l
13  chmod 7 file1.txt
14  ls -l
15  cat /etc/passwd
16  useradd mca
17  ls -l
18  groupadd mca3
19  ls -l
20  cat \etc\group
21  touch file3.txt
22  ls -l
23  chmod 2 file3.txt
24  ls -l
25  chmod 4 file3.txt
```

Chapter 5: Future Enhancements

- **Graphical User Interface (GUI):** A GUI-based user management tool for easier administration.
- **Integration with LDAP/Active Directory:** Integrate the system with existing enterprise identity management solutions.
- **Advanced Role-Based Access Control (RBAC):** Extend user roles to have more granular control over system resources.

Key Features of the System:

1. Automated User Creation:

- Admins can use a single command to create users with default settings (home directory, groups, shell, etc.).
- The system will call `useradd`, assign default permissions, and notify the admin once the user has been successfully created.

2. User Modification:

- The system will allow admins to modify user details (e.g., username, password, groups) via the `usermod` command.
- Any changes will be logged for auditing purposes.

3. User Deletion:

- When deleting a user, the system will offer the option to remove associated files (e.g., home directory, email files) using the `userdel` command.

4. Permission Management:

- Admins can assign and revoke permissions on specific files and directories using `chmod` and `chown` commands.
- Permissions can be set for users or groups, defining read, write, and execute rights on system resources.

5. Session Logging:

- The system will track user login attempts using Linux's `last` and `who` commands, storing login timestamps, IP addresses, and session durations.
- Reports will be generated showing login activity and failed attempts.

6. Security Management:

- Enforcing secure password policies (e.g., minimum password length, password expiration) through `/etc/login.defs`.

- Role-based access control (RBAC) could be incorporated for managing who can execute specific commands.

Implementation Steps:

1. Design Shell Scripts:

- Develop a set of shell scripts to handle user and group management tasks (create, modify, delete users).
- Scripts will use standard Linux commands but automate the process to reduce human error.

2. Integrate Permission Management:

- Implement functionality to assign and modify file permissions using chmod, chown, and setfacl (for access control lists).
- Provide easy-to-understand prompts for administrators to control access to critical directories.

3. Log User Activity:

- Capture system login activity, failed logins, and user commands in logs located in /var/log/.
- Use last, w, and who to track sessions and generate user activity reports.

4. Testing and Debugging:

- Test the system with various user management scenarios to ensure proper functionality.
- Use Linux virtual machines to simulate different environments (e.g., multi-user systems).

5. Security Measures:

- Secure user passwords and enable features like account lockouts after repeated failed login attempts.
- Implement error handling in the scripts to ensure that invalid operations do not compromise the system.

Chapter 6: Conclusion

The **User Management System** project will significantly improve the efficiency and security of Linux systems by automating user management tasks. The system will reduce administrative workload, minimize errors, and provide more reliable access control and auditing features. The system will be especially useful in large organizations or environments with multiple users, ensuring that system resources are properly allocated and protected.

Chapter 7: Bibliography

1. "Linux Command Line and Shell Scripting Bible" by Richard Blum and Christine Bresnahan.
2. "Linux System Administration" by Tom Adelstein and Bill Lubanovic.
3. "The Linux Documentation Project" (<https://www.tldp.org/>).
4. "Linux Security Cookbook" by Daniel J. Barrett.
5. "Advanced Bash-Scripting Guide" by Mendel Cooper (<https://tldp.org/LDP/abs/html/>).