Adventist University of Central Africa
P.O. Box 2461 Kigali, Rwanda | www.auca.ac.rw | info@auca.ac.rw

Faculty of Information Technology

Mid-Term Examination Academic Year: 2024-2025 (1)

Course Code & Name: COSC 8312 & Intro To Linux Administration

Lecturer: Joshua IRADUKUNDA Date: 28th October, 2024

MAX/30 **Group Day: (E)** Monday DURATION: 6 Hours

Instructions:

- ✓ *This exam consists of 3 questions. You must answer all of them.*
- ✓ *Each question is worth 10 points.*
- ✓ ***To submit**, please zip all files in your working directory and upload the compressed file. Ensure it includes your **Asciinema** recorded terminal file, scripts, and any relevant files and directories used in your work.*
- ✓ ***Avoid cheating or plagiarism; violations result in a zero, exam dismissal, and potential semester repeat.***

Question1: /10 points

Introduction

In this exercise, you will develop a Bash script that monitors a specified directory for any changes related to files and directories, logging these changes to designated log files. The script will also handle error conditions, ensuring that necessary directories and log files are created if they do not exist. You are required to incorporate your unique StudentID throughout the script to personalize your work.

This scenario aims to deepen your understanding of file monitoring in Linux environments and improve your scripting skills by applying real-world scenarios.

Objectives

By the end of this exercise, you should be able to:

1. Create a Bash script that monitors a specific directory for file and directory changes.
2. Log the changes in separate log files for directories, files, successful commands, and errors.
3. Implement error handling to ensure the script is robust and user-friendly.
4. Use your StudentID in the script to demonstrate originality.
5. Discuss the expected output of each component of the script.

Step-by-Step Guidance

1. Setting Up the Monitoring Environment:

- Begin by defining the directory you will monitor and the log directory where the logs will be stored. You should ensure that both the monitored directory and the log directory exist. If they do not, your script should create them automatically.
- When creating the directories, log the creation events, including your StudentID, to a setup log file. This logging allows you to keep track of your script's operations.

Example:

Your setup log may contain entries like:

```
2024-10-27 12:00:00 INFO: Created monitored directory /path/to/your/directory (StudentID: 123456)
2024-10-27 12:00:00 INFO: Created log directory /path/to/your/logs (StudentID: 123456)
```

---

2. Logging Directory Changes:

- Implement a mechanism to check for newly created or deleted directories within the monitored directory. For each directory event, log the directory's name and the type of event (creation or deletion) into a separate log file for directories.

Example:
Your directory log file may have entries like:

```
2024-10-27 12:05:00 DIRECTORY: created new_folder (StudentID: 123456)
2024-10-27 12:10:00 DIRECTORY: deleted old_folder (StudentID: 123456)
```

---

3. Logging File Changes:

- Similarly, you will need to monitor files in the specified directory. For every new file created, modified, or deleted, you should log the filename along with the event type in a separate log file dedicated to files.

Example:
Your file log file may include entries like:

```
2024-10-27 12:05:30 FILE: created new_file.txt (StudentID: 123456)
2024-10-27 12:15:00 FILE: modified existing_file.txt (StudentID: 123456)
2024-10-27 12:20:00 FILE: deleted unwanted_file.txt (StudentID: 123456)
```

---

4. Logging Command Success and Failures:

- Create functions to log the success and failure of commands executed within the monitored directory. For every successful command, log a success message that includes the command executed, and for every failure, log an error message with a description of the failure.

Example:

Your success log may show:

```
2024-10-27 12:25:00 SUCCESS: Executed command to create new_file.txt (StudentID: 123456)
```

Your error log may contain:

```
2024-10-27 12:30:00 ERROR: Failed to delete file that does not exist (StudentID: 123456)
```

---

5. Running the Monitoring Script:

- Once the script is implemented, ensure it runs in an infinite loop with a delay to check for changes every few seconds. The continuous nature of the script is crucial for real-time monitoring.
- Monitor the log files to verify that events are being captured correctly. Each log entry should have a timestamp and your StudentID for easy identification.

Expected Output/Outcomes

Upon successful implementation and execution of the script, you should expect to see:

- A structured log of all directory-related changes in the directory log file.
- A structured log of all file-related changes in the file log file.
- Separate logs for successful and failed command executions, each properly formatted with timestamps and your StudentID.
- A setup log indicating the creation of necessary directories and log files.

The outcome will demonstrate your ability to automate monitoring tasks using Bash scripting effectively while maintaining a clear and organized logging system.

Conclusion

By completing this exercise, you will have built a functional Bash script that monitors directory and file changes, enhancing your scripting skills and understanding of Linux file system operations. Remember to utilize your StudentID throughout your script to personalize your work and ensure its originality. This exercise not only solidifies your knowledge but also prepares you for practical applications in system administration and monitoring tasks.

Question2: /10 points

Introduction

In this practical examination, you will demonstrate your proficiency in user management and activity monitoring on a Linux system. The focus will be on creating user accounts, monitoring their login and logout activities, managing session times, and implementing logout policies based on inactivity. This exercise will also assess your understanding of how to log user activities and enforce policies that limit concurrent logins on a single host.

Objectives

By the end of this exam, you should be able to:

1. Create three user accounts with specific naming conventions that incorporate student IDs.
2. Monitor user login and logout times effectively.
3. Track user inactivity and implement a policy that forces users to log out after a set period of inactivity.
4. Limit the number of concurrent user logins to two and manage user sessions accordingly.

User Accounts Creation

You will create the following user accounts:

- **Masoro_studentID**
- **Gishushu_studentID**
- **Ngoma_studentID**

Ensure that the studentID portion of each username reflects your unique student identification number. For example, if your student ID is 12345, the usernames will be:

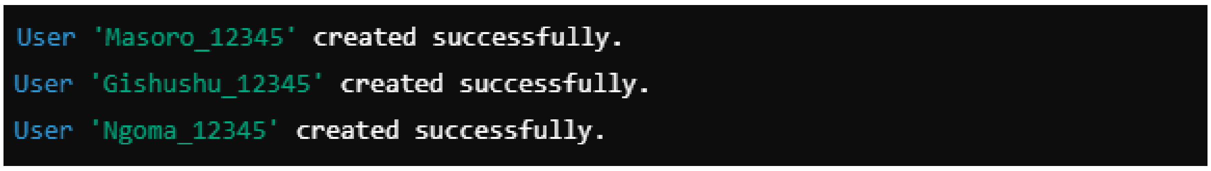
- **Masoro_12345**
- **Gishushu_12345**
- **Ngoma_12345**

Step-by-Step Guidance for Expected Output

1. **User Creation:**
    - You will begin by creating the three user accounts as specified. For each user account created, you should verify the successful creation by checking the user list or specific system files where user information is stored.

Expected Output Example:

- The user account for **Masoro_12345** is created successfully, followed by similar confirmations for the other two users.

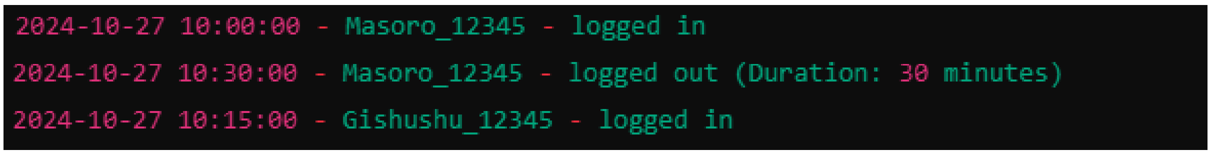


```
User 'Masoro_12345' created successfully.
User 'Gishushu_12345' created successfully.
User 'Ngoma_12345' created successfully.
```

2. **Monitoring Logins and Logouts:**
    - As users log in and out of the system, you will implement a logging mechanism to capture these activities. This log should include timestamps for when each user logs in and out, along with the duration of their session.

Expected Output Example:

- A log entry for **Masoro_12345** might display:

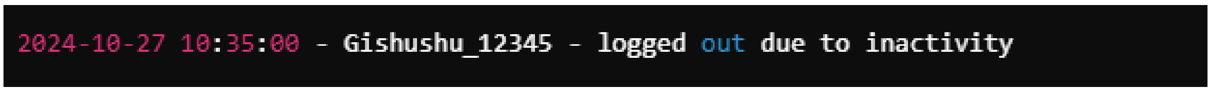


```
2024-10-27 10:00:00 - Masoro_12345 - logged in
2024-10-27 10:30:00 - Masoro_12345 - logged out (Duration: 30 minutes)
2024-10-27 10:15:00 - Gishushu_12345 - logged in
```

3. **Inactivity Monitoring:**
   - After the user accounts have been created and monitored for a period, you will set up a system to check for user inactivity. If any user is inactive for longer than the specified period, that user should be forcibly logged out of the system.

   Expected Output Example:

   - If **Gishushu_12345** has been inactive for over one minute, you will see:



```
2024-10-27 10:35:00 - Gishushu_12345 - logged out due to inactivity
```

4. **Limiting Concurrent Logins:**
   - You will then implement a policy to limit the number of concurrent logins to two users at any given time. If a third user attempts to log in, you must prompt the logged-in users to select one to log out.

   Expected Output Example:

   - Upon a third login attempt by **Ngoma_12345**, the system should display:

```
Too many users logged in. Prompting for logout selection.
Current logged-in users:
1) Masoro_12345
2) Gishushu_12345
Select the user number to log out:
```

5. **Final Logging and Reporting:**
   - After implementing the above functionalities, you will generate a final report summarizing all user activities, including their login/logout times, inactivity-related logouts, and any forced logouts due to concurrent login limitations.

   Expected Output Example:

   - The report might summarize activities as follows:

```
User Activity Report:
- Masoro_12345: Logged in at 10:00, logged out at 10:30 (Duration: 30 minutes)
- Gishushu_12345: Logged in at 10:15, logged out at 10:35 due to inactivity
- Ngoma_12345: Logged out to allow new login at 10:40
```

Conclusion

This practical exercise aims to solidify your understanding of user management in a Linux environment, focusing on creating user accounts, tracking activity, enforcing logout policies, and managing concurrent sessions. By successfully completing this exercise, you will demonstrate not only your technical skills but also your ability to enforce security and operational policies in a multi-user system. Ensure you document every step clearly, as this will reflect your understanding of the commands and processes involved.

Question3: /10 points

Introduction

In this practical exam, you will be tasked with creating a file management and monitoring system using a Bash script. The objective is to demonstrate your understanding of user management, file creation, directory manipulation, and process monitoring in a Linux environment. You will create three users, generate timestamped files in their respective directories, and implement a monitoring dashboard to track the created files. This exercise will also assess your ability to handle errors and implement best practices in scripting.

Objectives

By the end of this exercise, you will be able to:

1. Create user accounts with unique usernames based on a specified Student ID.
2. Implement a script that generates files with a specific naming convention every second for each user.
3. Move these files into designated user storage directories.
4. Monitor and display file counts and the latest files for each user.
5. Implement logic to archive files when a certain threshold is reached.

Task Instructions

1. **User Creation**:

- Create three users: Masoro_<StudentID>, Gishushu_<StudentID>, and Ngoma_<StudentID>. The <StudentID> should be unique and should appear in all relevant file and directory names.
- Ensure that the home directories for each user exist and are correctly configured to allow file manipulation.

2. **Directory Structure**:
   - For each user, create a dedicated storage directory in their home directory named **<Username>_Storage_<StudentID>**.
   - Additionally, create a common archive directory named **/home/Masoro/Archives** to store archived files.
3. **File Generation Logic**:
   - Implement a loop in your script that runs indefinitely to create files every second. The files should follow the naming convention **File_<Username>_<StudentID>_<Timestamp>.txt**, where **<Timestamp>** is in the format **YYYYMMDD_HHMMSS**.
   - Each user's file should be generated in the respective user storage directory.
4. **File Movement**:
   - Move the generated files into the respective user storage directories immediately after creation.
5. **Monitoring Dashboard**:
   - Create a dashboard that displays the following information:
     - The total number of files in each user's storage directory.
     - The name of the most recently created file for each user.
   - The dashboard should update every few seconds to reflect the current state of the file system.
6. **Archiving Logic**:
   - Implement a mechanism to count files in each user's directory. When the file count reaches 10, move the oldest file to the archive directory /home/Masoro/Archives.

- Ensure that the archive directory is monitored and displays how many files are stored, as well as the name of the latest archived file.

Command May Need (Reference Only)

Although this section is not to be answered, the following commands may be useful for your implementation:

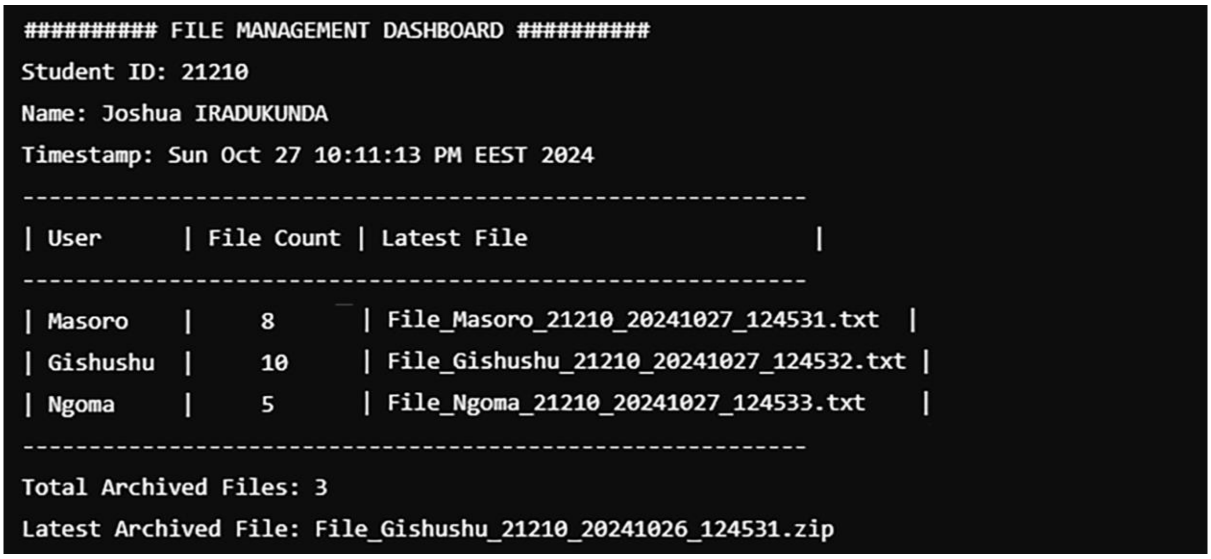
- User and directory creation commands.
- File manipulation commands (creation, moving).
- Commands for listing files and counting them.
- Monitoring processes and displaying outputs in a structured format.

Expected Output / Outcomes

The expected outcome of your task is as follows:

1. **User Creation**:
   - Upon successful user creation, you should be able to see the three users listed in the system.
   - Each user should have a home directory containing the respective storage directory.
2. **File Creation**:
   - The script should generate files in the format File_Masoro_<StudentID>_<Timestamp>.txt, File_Gishushu_<StudentID>_<Timestamp>.txt, and File_Ngoma_<StudentID>_<Timestamp>.txt every second, with the appropriate timestamps.
3. **Dashboard Monitoring**:
   - The dashboard should clearly present the number of files for each user and the latest file created in their directory.

- The output should refresh at a regular interval, reflecting real-time changes in the file system.



```
########## FILE MANAGEMENT DASHBOARD ##########
Student ID: 21210
Name: Joshua IRADUKUNDA
Timestamp: Sun Oct 27 10:11:13 PM EEST 2024
-----------------------------------------------------------
| User      | File Count | Latest File                      |
-----------------------------------------------------------
| Masoro    |     8      | File_Masoro_21210_20241027_124531.txt  |
| Gishushu  |     10     | File_Gishushu_21210_20241027_124532.txt |
| Ngoma     |     5      | File_Ngoma_21210_20241027_124533.txt    |
-----------------------------------------------------------
Total Archived Files: 3
Latest Archived File: File_Gishushu_21210_20241026_124531.zip
```

4. **Archiving Mechanism**:
   - Once a user's storage directory reaches 10 files, the oldest file should be moved to the archive directory, and the dashboard should update to reflect this action.
5. **Error Handling**:
   - The script should handle errors gracefully, informing the user of any issues that arise during execution, such as failures in file creation or movement.

Conclusion

In this exercise, you are expected to showcase your ability to work with user accounts, file systems, and Bash scripting. This practical exam will test not only your technical skills but also your creativity and ability to produce a well-structured and functional solution. Take the time to review your work and ensure all components are functioning as expected. Aim to implement best practices in scripting to produce a robust and efficient solution.

Additional Discussion Points

- Consider how user permissions might affect your script's execution. Will your script run under different user accounts?
- Think about how you could enhance the monitoring dashboard with additional statistics or visual elements.
- Explore the use of functions in your script to organize code better and enhance readability.
- Reflect on potential edge cases (e.g., what happens if the user directories do not exist at the start) and how you can handle them in your script.

Good Luck!!!