Ahmedabad University

Course: CSE404 Operating Systems Lab
Assignment 3: File Operations

Instructions

- Attempt **All Questions**.
- Submit a single **PDF** with shell script code followed by terminal output screenshots.
- Filename format: RollNo Name Assignment3.pdf

Question 1. Let us begin the exercise with the easiest task: Write a shell script that prompts the user to enter a folder name. The script should then create this folder and, inside it, create four empty Python files named 1.py, 2.py, 3.py, and 4.py.

Hint: Use command *"mkdir"* for making the folder and *"touch"* for making the files

Question 2. Write a shell script that prompts the user to enter a file name. The script should then create an empty text file with the given name in the current directory and write *Hello Shell* into the .txt file through the terminal.

Hint: Use command *"touch"* to create an empty file.

Question 3. Write a shell script that creates a directory named backup and then creates copies of the files created in the Directory made in Question-1 to the backup directory name it log1.py, log2.py, log3.py and log4.py.

Question 4. Write a shell script that makes Directory named *database* and makes files named *pwdfile.txt*, *username.txt* into the directory

Question 5. Task: In Question 1, we created a Python file named 1.py using a shell script.
Write a shell script that prompts the user to enter any code (for example, *print("Hello Shell")*). The script should write the entered code into the 1.py file and then display the contents of the 1.py file on the terminal. The key goal is to practice displaying the file's contents after writing to it.

Question 6. Write a shell script which scans through all the files present in the directory made in Question-1 and list down the file name which has the string *Hello* in it.

Question 7. Write a shell script that creates a directory named temp in the current direc- tory, navigates into it, creates another directory named test inside temp, then navigates back to the parent directory and deletes the temp directory.

Question 8. Write a shell script that prompts the user to enter a file name and then checks if the file exists in the current directory. If it exists, display its contents; otherwise, display a message saying the file does not exist.

ShellBank File Operations Management

You are a system administrator at **ShellBank**, a financial institution that manages customer accounts using a Linux-based system. The bank stores customer data in a directory called database, created in Question 4 , which contains username.txt and pwdfile.txt. Your task is to write shell scripts to manage customer accounts, handle file operations, and ensure data integrity. The following question builds on this scenario and requires you to use shell scripting commands to perform specific tasks.

Question 9. The following tasks involve managing customer data in the database direc- tory. Write shell scripts for each subtask below, ensuring proper navigation, file handling, and error checking. You may create additional files or directories as needed to complete the tasks.

- **Subtask 1: Create a new customer account**

  Write a shell script that navigates to the database directory and prompts the user to enter their username and password. The script should append the username to username.txt, the password to pwdfile.txt, and create a new file balance.txt with an initial balance of 0.
  Items to Create: Create one file: balance.txt in the database directory; append to existing username.txt and pwdfile.txt.

- **Subtask 2: Verify customer account details**

  Write a shell script that navigates to the database directory and displays the contents of username.txt, pwdfile.txt, and balance.txt on the terminal. Ensure the script checks if these files exist before displaying their contents, and shows an error message if any file is missing.
  Items to Create: Create zero files.

- **Subtask 3: Update customer account balance**

  Write a shell script that navigates to the database directory, prompts the user to en- ter a new balance, and overwrites the balance.txt file with this new amount. After updating, display the updated balance using a command that shows the file contents page by page.
  Items to Create: Overwrite one file: balance.txt in the database directory.

- **Subtask 4: Organize customer data into separate directories**

  Write a shell script that prompts the user to enter a customer's username, creates a new directory with that username inside the database directory, and moves the username.txt, pwdfile.txt, and balance.txt files into this new directory.
  Items to Create: Create one directory: <username> in the database directory.

  Number of Directories: Create one directory.

Question 10.

Subtask 5: Log File Cleanup Script

Write a shell script that searches for all .log files in a given directory and its subdirectories. The script should delete files that are older than 7 days and generate a summary report of deleted files in a file named deleted logs report.txt.

Question 11.

Subtask 6: File Difference Detector

Write a shell script that takes two filenames as input from the user. It should compare the files line by line and display only the differing lines along with their respective line numbers from each file.

Question 12.

Subtask 7: Batch Rename Files

Write a shell script that takes a directory name as input and renames all .txt files in that directory by adding the prefix backup to each file. The script should print a list of the original and renamed filenames.

Question 13.

Subtask 8: File Content Merger and Sorter

Write a shell script that accepts two text files as input, merges their content into a third file, sorts the merged content alphabetically, and removes duplicate lines. Display the final output in the terminal.

Question 14.

Subtask 9: Directory Disk Usage Reporter

Write a shell script that takes a directory path as input, recursively calculates the size (in MB) of each file within it, and displays a sorted list of files from largest to smallest along with their sizes.

Possible Commands Used

The following commands may be useful for solving the above questions:

- cd: Change directory.
- mkdir: Create a directory.
- rmdir: Remove an empty directory.
- cat: Display file contents.
- cp: Copy files or directories.
- rm: Remove files or directories.
- mv: Move or rename files or directories.
- more: Display file contents page by page.
- file: Determine file type.
- wc: Count lines, words, and characters in a file.
- cmp: Compare two files byte by byte.
- diff: Show differences between two files.
- tar: Archive and compress files.

Urban City’s Traffic File Management

You are a data engineer for **Urban City’s traffic management system**, responsible for managing sensor data in a Linux-based traffic logs directory. You are provided with the traffic logs directory, which contains one subdirectory, Traffic data, hold- ing a single file, sensor.txt. The sensor.txt file contains traffic metrics with lines like 2025-06-24 08:00,150, where the first part is a timestamp and the second is a vehicle count; negative counts (e.g., -5) indicate corrupted data. A sudden surge in traffic has overwhelmed the system, causing data issues like corruption and duplication. Your task is to write advanced shell scripts to process, analyze, and archive the data in sensor.txt to restore system stability and prevent gridlock. You may create additional files or directories as needed to complete the tasks.

Question 14. Task: The traffic surge corrupted data in the provided sensor.txt file. You are given the traffic logs directory with a Traffic data subdirectory containing sensor.txt. Write a shell script that navigates to the traffic logs/Traffic data di- rectory, checks sensor.txt for corrupted entries (negative vehicle counts) using grep, and creates a report file corruption_report.txt listing all corrupted lines and the total number of corrupted entries using wc. Display the report using more.
Provided Data: traffic logs/Traffic data/sensor.txt with traffic metrics.
Items to Create: Create one file: corruption_report.txt in the traffic logs/Traffic data directory.
Number of Directories: Create zero directories.
Data Hint: sensor.txt has lines like 2025-06-24 08:00,150. Negative counts (e.g., -5) are corrupted.

Question 15. Task: The city needs to split sensor.txt into hourly summaries for analy- sis.
You are given the traffic logs directory with a Traffic data subdirectory containing sensor.txt. Write a shell script that navigates to the traffic logs/Traffic data direc- tory, uses grep to extract lines from sensor.txt for a user-specified hour (e.g., 2025-06-24 08), and saves them to a new file hourly_summary.txt. Count the total vehicle count for that hour using awk and append it to hourly_summary.txt. Display the file contents using cat.
Provided Data: traffic logs/Traffic data/sensor.txt with traffic metrics.
Items to Create: Create one file: hourly_summary.txt in the traffic logs/Traffic data directory.

Question 16. Task: Archive the provided sensor.txt data to save storage.
You are given the traffic logs directory with a Traffic data subdirectory containing sensor.txt. Write a shell script that navigates to the traffic logs directory, creates an archive directory, compresses Traffic data/sensor.txt into traffic_data.tar.gz

using tar, and moves it to the archive directory. Verify the archive by extracting it to a temporary temp directory and comparing sensor.txt with the original using cmp. Remove the temp directory afterward.
Provided Data: traffic logs/Traffic data/sensor.txt with traffic metrics.

Items to Create: Create one file: traffic_data.tar.gz in the traffic logs/archive

directory; create two directories: archive and temp in the traffic logs directory.

Data Hint: sensor.txt has lines like 2025-06-24 08:00,150.

Question 17. Task: Clean up invalid data in the provided sensor.txt file.

You are given the traffic logs directory with a Traffic data subdirectory containing sensor.txt. Write a shell script that navigates to the traffic logs/Traffic data direc- tory, uses sed to remove all lines with negative vehicle counts from sensor.txt, saving the cleaned data to cleaned_sensor.txt. Log the number of removed lines to cleanup_log.txt using wc, and verify the cleaned file by displaying its contents using more.
Provided Data: traffic logs/Traffic data/sensor.txt with traffic metrics.

Items to Create: Create two files: cleaned_sensor.txt and cleanup_log.txt in the

traffic logs/Traffic data directory.

Number of Directories: Create zero directories.

Data Hint: sensor.txt has lines like 2025-06-24 08:00,150. Negative counts (e.g., -5) are invalid.

Question 18. Task: Detect and remove duplicate entries in the provided sensor.txt file. You are given the traffic logs directory with a Traffic data subdirectory containing sensor.txt. Write a shell script that navigates to the traffic logs/Traffic data di- rectory, checks for duplicate lines in sensor.txt using sort and uniq, and creates a new file unique_sensor.txt with duplicates removed. Log the number of duplicates found to duplicate_log.txt using wc, and compare the original and new files using diff. Display the log using cat.
Provided Data: traffic logs/Traffic data/sensor.txt with traffic metrics.

Items to Create: Create two files: unique_sensor.txt and duplicate_log.txt in the

traffic logs/Traffic data directory.

Number of Directories: Create zero directories.

Data Hint: sensor.txt has lines like 2025-06-24 08:00,150. Duplicates are identical lines.

Possible Commands Used

The following commands may be useful for solving the above questions:

- cd: Change directory.
- mkdir: Create a directory.

- rm: Remove files or directories.
- mv: Move or rename files.
- cat: Display or concatenate file contents.
- more: Display file contents page by page.
- wc: Count lines, words, and characters.
- cmp: Compare two files byte by byte.
- diff: Show differences between files.
- tar: Archive and compress files.
- gzip, gunzip: Compress or decompress files.
- find: Locate files in a directory hierarchy.
- grep: Search for patterns in files.
- sed: Stream editor for text transformation.
- sort: Sort lines of text files.
- uniq: Remove duplicate lines from sorted files.
- awk: Process and analyze text data.