Canterbury Christ Church University

SCHOOL OF ENGINEERING, TECHNOLOGY & DESIGN ASSIGNMENT GUIDELINES

TITLE OF MODULE: Advanced Operating Systems (U14553)	**MODULE COMPONENT:** **50% of Module**
MODULE TEAM: Victor Obarafor Merlin Kasirajan Vijay Sahota Tim Jackson	**ASSIGNMENT CONTACT**: Vicor Obarafor Victor.obarafor@canterbury.ac.uk
ASSIGNMENT DEADLINE: 24th Mar 14:00 2025	**EXPECTED FEEDBACK DATE:** 10th Apr 2025 **Location of Feedback:** VIA TURNITIN ON BLACKBOARD
ASSESSMENT TYPE	Report + Artefact
WHERE TO SUBMIT: BLACKBOARD TURNITIN SUBMISSION TOOL If you experience any problems with this system, then please contact the Computing Administration Team (computing@canterbury.ac.uk) **WHAT TO SUBMIT:** **A 800 (Max) word report of high professional standard + Artefact source code**	

ASSIGNMENT INSTRUCTIONS

This assessment evaluates your ability to apply operating system management techniques and automation tools in a real-world context. It is structured into three scripting-based tasks, requiring you to write Bash or Python scripts to manage files, process queues, and handle automated submissions.

1. University File Management & Automated Backup System (Bash Scripting Only). This task aligns with learning outcome 2, to demonstrates system automation techniques for file management, backups, and log tracking, key tools for OS administration.

The IT Services Department at Christ Church University is responsible for maintaining critical administrative files, research documents, and student records. However, due to frequent manual errors and lost files, the university requires an automated file management system that allows IT staff to navigate directories, manage files, and execute automatic backups.

Your job is to write a Bash script that simulates an intelligent file manager, capable of executing the following operations via a menu-driven interface:

Requirements

1. Directory Navigation & File Management

    - List all files in a directory with details (name, size, last modified date).
    - Move a file to a different folder, ensuring the destination exists.
    - Rename files while checking for naming conflicts (prevent overwriting).
    - Delete files permanently after asking for a confirmation prompt (Y/N).

2. Automated Backup Functionality

    - Create a Backup/ directory (if it does not exist).
    - Copy files to the Backup/ folder with a timestamped filename.
    - If the Backup/ folder exceeds 500MB, trigger a warning.

3. Logging System

    - Maintain a backup_log.txt that tracks every operation, including timestamps.

4. Exit Mechanism

    - Implement a "Bye" exit command that prompts users for confirmation (Y/N).

2. Christ Church University Library Smart Borrowing System (Bash or Python) . This task aligns with learning outcome 4 which requires using Bash/Python scripting to manage a queue system efficiently, reinforcing POSIX standard concepts.

The Christ Church University Library is experiencing a major issue: students frequently request books, but due to high demand, some books are either unavailable or students unfairly skip the queue. To fix this, the university wants an automated system that processes book borrowing requests using either FIFO (First-In-First-Out) or Priority Scheduling.

As a systems administrator, your job is to develop a Bash or Python script that handles real-time book requests and processing.

Requirements

1. User Menu

    - View available books in the library.
    - Request a book (add to the queue).
    - Process book requests using either:
    - FIFO (First Come, First Served)

- Priority Scheduling (Students with higher priority receive books first).
- Exit system (with a confirmation prompt).

2. Queue-Based Borrowing System

   - Students can request a book with an optional priority number (1-10).
   - FIFO should serve students in order of request time.
   - Priority scheduling should serve students with the highest priority first.

3. Logging System

   - Log every book request and lending in library_log.txt.
   - Keep track of students who borrowed books and the order in which they were served.

4. Data Storage

   - Store book requests in book_requests.txt

3. University Examination Submission & Similarity Detection (Bash & Python).

Task 3 Aligns with the learning outcomes 2 and 4. Demonstrates system administration techniques for managing file validation, storage, and duplicate detection and applying Bash and Python scripting for handling file management, format validation, and automated similarity checks.

The Examination Board of Christ Church University has implemented a new automated submission system that ensures students submit unique assignments while preventing plagiarism and duplicate submissions.

Your task is to develop a script that validates and tracks student submissions.

Requirements

1. Menu System

   - Submit an assignment.
   - Check if a file has already been submitted.
   - List all submitted assignments.
   - Exit the system (with confirmation).

2. File Validation

   - Only accept .pdf and .docx files.
   - Reject files larger than 5MB.

3. Duplicate Submission Detection

    - If a student submits a file with the same name and content, reject it.
    - Log all submissions in submission_log.txt.

4. Plagiarism Detection (Optional - Bonus Marks)

    - Use Python to scan text for similarity between assignments.
    - If similarity is above 90%, flag it.

Report Write-Up (800 Words) Research & Justification

Alongside your source code, you must submit an 800-word report that explains your implementation in detail. This report should:

1. Explain Your Design Choices:

    - Justify the use of specific Bash or Python functions in your scripts.
    - Compare alternative methods and explain why you selected your approach.

2. Connect to Advanced Operating System Concepts:

    - Reference relevant concepts from the module, such as process management, scheduling algorithms, file system operations, or automation techniques.
    - Discuss how your task solutions align with real-world system administration practices.

3. Critical Research & Referencing:

    - Support your explanations with at least three academic or technical sources (e.g., textbooks, research papers, official documentation).
    - Harvard Referencing must be used for all sources.
    - Plagiarism is strictly prohibited, and all references must be cited properly.

4. Challenges & Solutions:

    - Identify any difficulties encountered and explain how you resolved them.
    - Reflect on what could be improved in your implementation.

5. Execution Steps & Expected Output:

    - Provide a clear step-by-step guide on how to execute each script.
    - Include sample outputs/screenshots to demonstrate correctness.

MARKING SCHEME

Task	Marks
Task 1: File Management & Backup	15
Task 2: Library Queue System	15
Task 3: Exam Submission System	15
Report Write-up	5
TOTAL	50

ASSESSMENT RUBRIC

Each task is graded out of 15 marks, while the report write-up is worth 5 marks, making a total of 50 marks.

The final grade is determined by the sum of all three tasks and the report.

Task 1 University File Management & Automated Backup System (15 Marks)

Criteria	Marks	Description
Menu System & User Interaction	3	The user can navigate directories through an interactive menu. Includes all necessary options (list, move, rename, delete, backup, exit). Handles invalid inputs gracefully.
File Operations (Move, Rename, Delete)	4	Move files correctly, ensuring the target folder exists. Rename files while checking for naming conflicts. Deletes files permanently only after a confirmation prompt (Y/N).
Automated Backup System	4	Creates a Backup/ folder and saves files with a timestamped filename. If the backup folder exceeds 500MB, displays a warning. Files are copied securely, and original files remain intact.
Logging System	2	Every file operation (move, delete, rename, backup) is recorded in backup_log.txt. Logs contain timestamps and are structured clearly.

Exit Mechanism & Confirmation	2	Proper ‘Bye’ command with a Y/N confirmation prompt before exiting.

Task 2 Library Queue System – Smart Borrowing System (15 Marks)

Criteria	Marks	Description
Menu System & User Interaction	3	Users can navigate through options (view books, request books, process queue, exit). Proper error handling for invalid inputs.
Book Request Processing (FIFO & Priority)	4	FIFO implementation processes requests in the order they were made. Priority scheduling correctly serves students with higher priority values first. System ensures students can only request books that exist in the library.
Queue & Data Storage System	4	Requests are stored in book_requests.txt. Completed book requests are properly removed from the queue. Queue structure is correctly implemented and sorted (for Priority scheduling).
Logging System	2	Every book request and processed loan is recorded in library_log.txt. Logs include timestamps, student ID, book title, and queue type (FIFO/Priority).
Exit Mechanism & Confirmation	2	- Proper ‘Bye’ command with a Y/N confirmation prompt before exiting.

Task 3 Examination Submission & Similarity Detection System (15 Marks)

Criteria	Marks	Description
Menu System & User Interaction	3	Users can navigate through the menu (submit assignment, check existing submissions, list submissions, exit). Proper error handling for invalid inputs.
File Validation (Size & Format)	4	System only accepts .pdf and .docx files. Files larger than 5MB are rejected. Invalid formats are immediately flagged.
Duplicate Submission Detection	4	The system checks for duplicate filenames and content before accepting a submission. If the content is the same, the system rejects the file and prompts the user.
Logging System	2	Every submission is recorded in submission_log.txt. Logs include student ID, filename, timestamp, and duplicate status.

Exit Mechanism & Confirmation	2	Proper ‘Bye’ command with a Y/N confirmation prompt before exiting.

Report Write-Up (5 Marks)

Criteria	Marks	Description
Code Explanation & Screenshots	2	Student provides clear explanations of their code, including screenshots of execution.
Challenges & Solutions	1	Describes difficulties faced and how they were resolved.
Execution Steps & Expected Output	1	Explains how the script is executed and what output to expect.
Professional Formatting & Clarity	1	Well-structured, well-written, and free of grammatical errors.

Submission Guidelines – Christ Church University Advanced Operating Systems 2025

File Naming and Turnitin Submission

Turnitin submissions are now anonymized. To ensure your source code and written report can be accurately matched, you must follow the file naming convention below for your code submission:

File Naming Format:

Your ZIP file should be named using the following calculation:

1. Take the day of your birth (e.g., if you were born on the 5th, use 5).
2. Add the month of your birth as a number (e.g., January = 1, February = 2, etc.).
3. Add the full year of your birth (e.g., 2025).
4. Multiply the sum by a randomly chosen number between 111 and 999.

Example Calculation:

- Date of Birth: 1st January 2025
- Step 1: 1 (day) + 1 (month) + 2025 (year) = 2027
- Step 2: Multiply by a random number, e.g., 234
- Final ZIP Name: 2027 x 234 = 474318.zip

Your written submission must clearly state the exact ZIP filename you have used, so it can be accurately linked to your code submission.

Submission Instructions

1. Upload your written report (.docx or PDF) onto Blackboard (Turnitin).

    - A submission bucket/link will be provided in the same location where you downloaded this assignment.

2. Upload your ZIP file (containing all source code) onto Blackboard (Turnitin).

    - Ensure all your scripts are inside this one ZIP file, following the specified naming format.

Important Submission Rules

- Failure to submit a written report will result in a ZERO mark.
- Failure to provide an adequate write-up will result in mark scaling (penalty).
- Your code will be tested by your instructor

Strictly Prohibited

- Use of Awk, Sed, or built-in menu generators is NOT allowed.
- If used, you will receive ZERO marks for the affected sub-task(s).

For further details on assessment criteria, refer to the mark scheme provided.

Formatting of Submission

- You are to upload your written report & source code onto Blackboard (Turnitin).

    - A dedicated submission bucket/link will be provided in the same location where you downloaded this assignment.

- Any screenshots, tables, figures, charts, illustrations, etc., will not contribute toward the word count.
- Your work must be adequately referenced throughout, following the Harvard Referencing Style.

    - Guidelines on using the Harvard Referencing style are available at: Harvard Referencing Guide.

- The report must be submitted using the dedicated Blackboard grade centre submission bucket on or before the submission deadline.

LEARNING OUTCOMES ASSESSED (Fully or Partially):

Learning Outcomes of this module:

1) **Outcome 2:** Exploit tools and techniques that allow administrators to manage the operating system efficiently and critically evaluate their efficacy.
2) **Outcome 4:** Demonstrate the ability to competently operate the UNIX command-line environment, including knowledge of the POSIX standard and advanced shell scripting.

GRADUATE / EMPLOYABILITY SKILLS GAINED:

This assessment is an opportunity to develop an understanding of the advance bash, their respective tool at hand and how to use them for complex tasks/ operations.

All tasks build on skills and knowledge required in industry, namely researching a problem (analysis), developing an idea (conception/ design/ communication) and documentation (though informal).

This assessment has been designed to exercise your abilities to work on your own and progress closer to becoming a proficient self-learner.

PROGRAMMES OF STUDY:

BSc (Hons) Computing & All routes.