Question_1 Report

Script Overview:

The provided shell script, named server_monitor.sh, automates the management and monitoring of server resources.

Given Tasks:-

1. **Disk Usage Monitoring:**

   Monitors disk usage and generates an alert if it exceeds a specified threshold (default: 30%).

2. **CPU Usage Monitoring:**

   Monitors CPU usage and generates an alert if it exceeds a specified threshold (default: 30%).

3. **Memory Usage Monitoring:**

   Monitors memory (RAM) usage and generates an alert if available memory falls below a specified threshold (default: 10% free).

4. **Log Rotation:**

   Implements log rotation for a specified log file to prevent it from growing beyond a specified size (default: 10MB).

Script Structure:

Log Rotation Function:

- This function check the log file exceeds the limit.

- Create new file if rotation is required and change the current file of log to **.old**

Main portion of script:

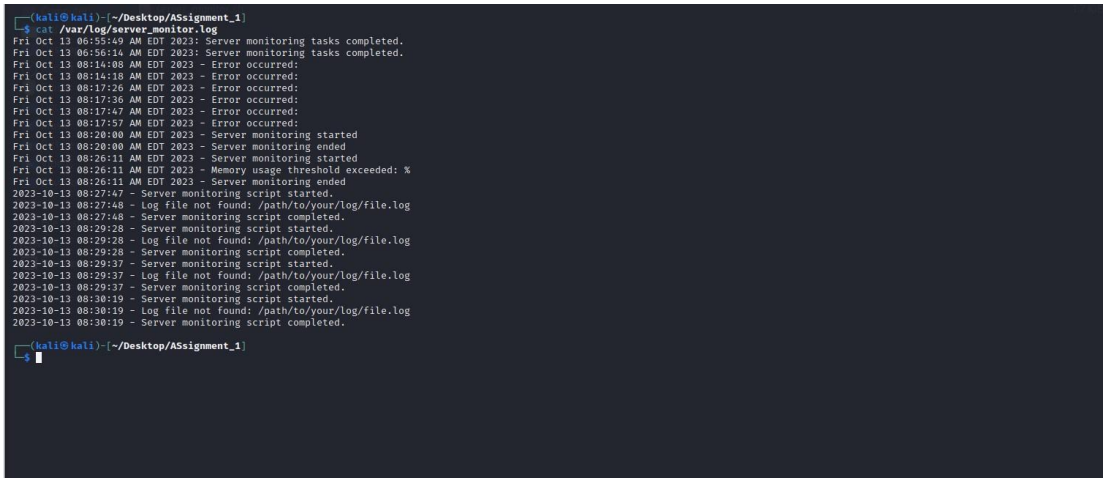
- Creating log file.
- Functions calls for Disk, Memory, and CPU.
- Maintaining the information of script start and end time

How to use script:

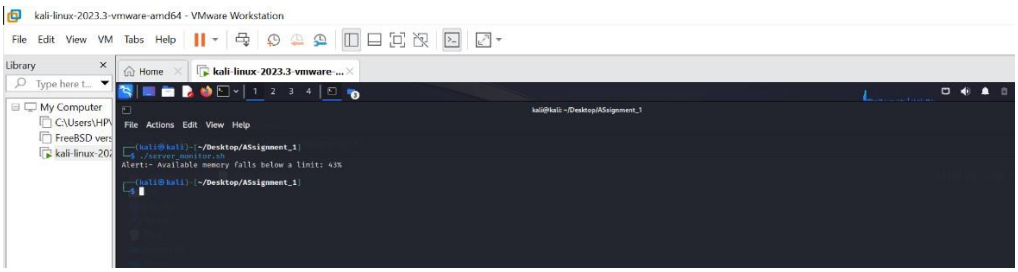
- Use Linux server to execute the script.
- Check the log file to monitor resources, any alerts script start end time and errors.

Output:

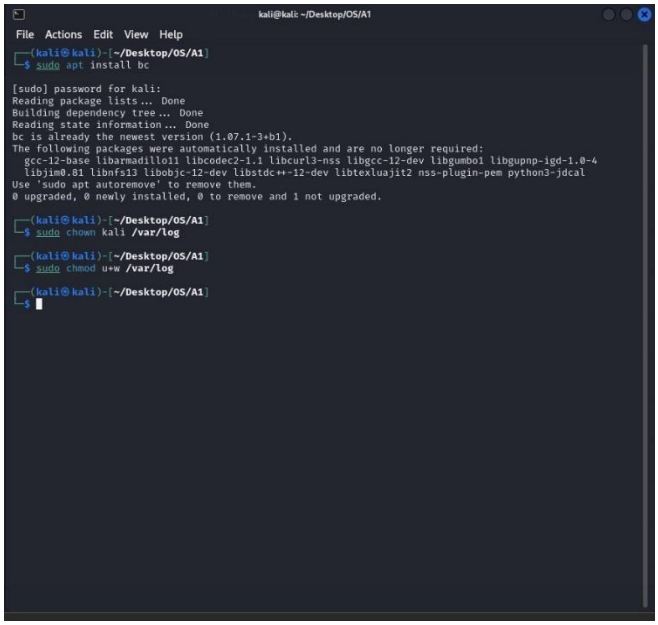
This script is not showing directly output to the file except alerts. To review logs you must use following command:



- Also execute your file to check there is any alert



- If your file is not executing then try this:



Conclusion:

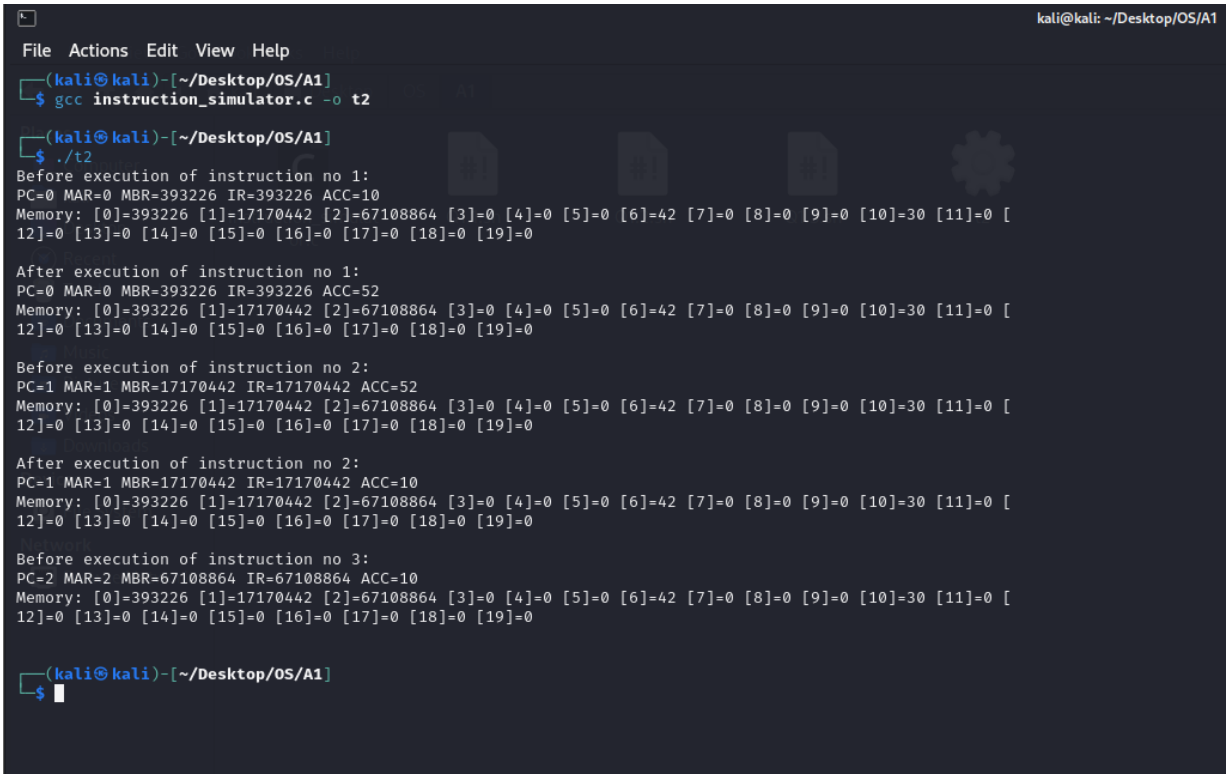
The server_monitor.sh script provides basic server resource monitoring and log rotation functionality.

Question_2 Report

Compilation and Running Commands:

1. gcc instruction_simulator.c -o t2
2. ./t2

Output :



Explanation of code:

- An enum named opcode and defines operations like ADD, SUB, LOAD, STORE and HALT.
- Variables of registers and memory. Memory is an array of size 20 to store memory location of data and instruction. Variables to cater the number of instruction is also made.
- Random instructions are loaded in memory to avoid null output.

- For Fetch ,decode and execute cycle a while loop is used that runs till the end of instructions.
- First fetch the next instruction and load it in PC register as memory address.
- Print the memory variable and all registers before the execution of instruction.
- Gets opcode and operands which are actual the data from IR.
- Checks if the instructions opcode is HALT the loop breaks.
- Next a switch is used that checks the opcode of instruction and execute according to it.
- Print the memory and registers after execution of instruction.
- Increment PC and the number of instruction.
- When the instructions finish the loop exits and return 0 is executed.

Conclusion:

The above code explains the fundamental principles of instruction execution in computer system and how memory and registers change throughout the process.