In Memory File System
Vijay 24CS60R40

Command.cpp

Data Structures:

- **Superblock:**
  - Stores metadata about the file system, including the total number of inodes and the number of free inodes.
- **Inode:**
  - Represents a file or directory.
  - Stores metadata like filename, size, type, permissions, and a pointer to the data block.
- **File Table:**
  - Keeps track of open files, including the file pointer, mode of access, and a reference to the corresponding inode.
- **UFDT (User File Descriptor Table):**
  - An array of file table pointers, used to map file descriptors to file table entries.

Functions:

- **CreateDILB:**
  - Initializes the inode list (DILB) with a fixed number of inodes.
- **InitaliseSuperBlock:**
  - Initializes the superblock and UFDT.
- **CreateFile:**
  - Allocates a free inode.
  - Initializes the inode's metadata.
  - Adds an entry to the UFDT.
- **rm_File:**
  - Decrements the link count of the inode.
  - If the link count reaches zero, the inode and its data block are freed.
- **ReadFile:**
  - Reads data from the file's data block.
  - Updates the file pointer.
- **WriteFile:**
  - Writes data to the file's data block.
  - Updates the file size and modification time.
- **CloseFileByName:**
  - Decrements the reference count of the file table entry.
- **ls_file_in_detail:**
  - Lists files with detailed information.
- **ls_file:**
  - Lists file names.

Shell.cpp

Functions:

- **sigintHandler:**
  - Handles the SIGINT signal (Ctrl+C) to prevent termination.
- **threadCreateFile, threadWriteFile, threadReadFile, threadDeleteFile:**
  - Helper functions for multi-threaded execution of file operations.
- **Test:**
  - Creates, writes to, reads from, and deletes multiple files in a multi-threaded manner.
  - Measures the execution time of each operation.
- **WrapperTest:**
  - Calls the `Test` function multiple times with different numbers of files.
  - Calculates the average latency for each operation.

Logic Explanation:

1. **File System Initialization:**
   - The `InitaliseSuperBlock` and `CreateDILB` functions initialize the basic data structures of the file system.
2. **File Creation:**
   - The `CreateFile` function allocates a free inode from the inode list and initializes its metadata.
   - An entry is added to the UFDT to track the open file.
3. **File Deletion:**
   - The `rm_File` function decrements the link count of the inode.
   - If the link count reaches zero, the inode and its data block are freed.
4. **File Access:**
   - The `ReadFile` and `WriteFile` functions access the file's data block through the inode and file table.
5. **File Listing:**
   - The `ls_file` and `ls_file_in_detail` functions iterate through the inode list to display file information.
6. **Multi-Threaded Testing:**
   - The `Test` function creates multiple threads to perform file operations concurrently.
   - This helps evaluate the performance and scalability of the file system.

Key Points:

- The file system uses a simple inode-based structure to manage files.
- The UFDT is used to track open files and their corresponding inodes.
- The multi-threaded testing framework allows for performance evaluation under different workloads.
- The code includes basic error handling and synchronization mechanisms.

GenerateReport.cpp

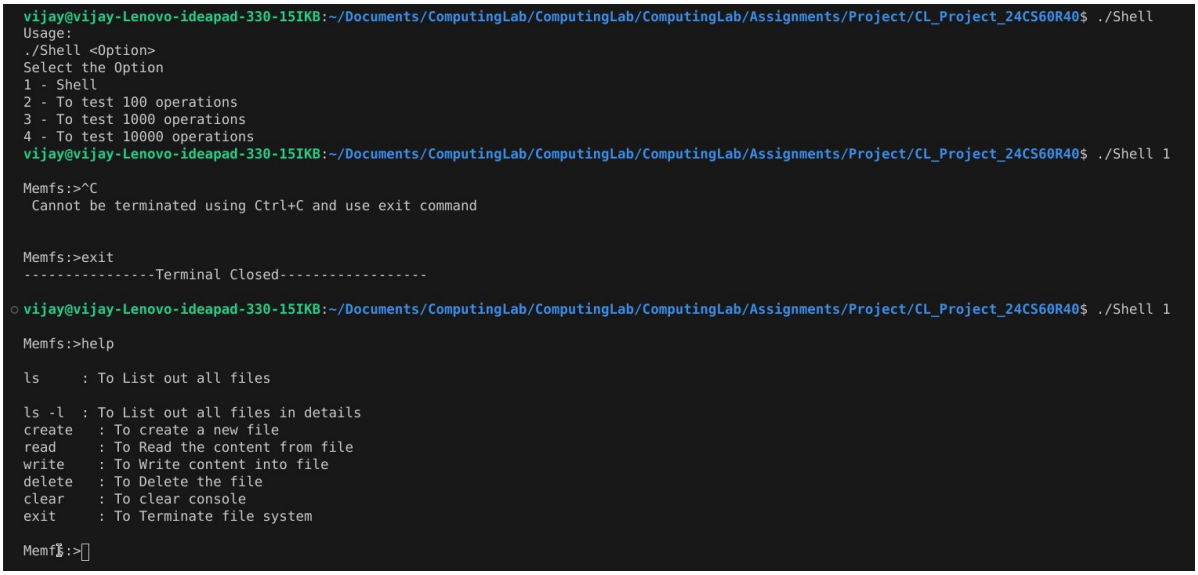
Functions:

1. **`getMemoryUtilization()`:**
    - **Purpose:** Calculates the current memory utilization of the system.
    - **Steps:**
        1. Retrieves system memory information using `sysinfo()`.
        2. Calculates the total memory, free memory, and used memory.
        3. Returns the used memory as a percentage of the total memory.
2. **`getCpuUtilization()`:**
    - **Purpose:** Calculates the CPU utilization over a short interval.
    - **Steps:**
        1. Reads CPU usage statistics from `/proc/stat`.
        2. Calculates the difference in idle and total CPU time between two consecutive readings.
        3. Returns the CPU utilization as a percentage.

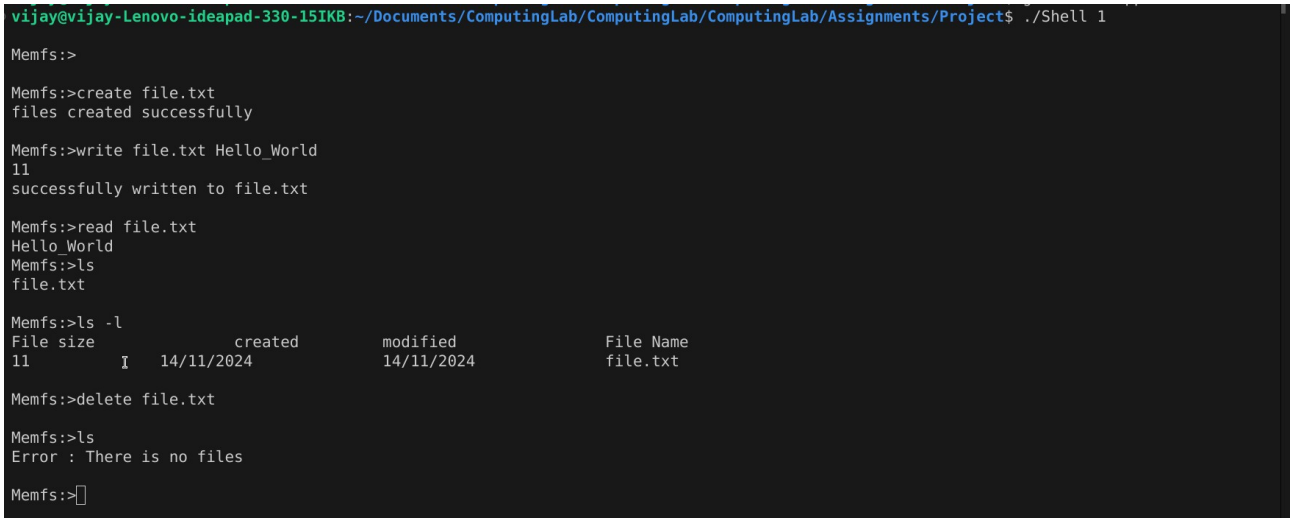
Main Function:

1. **Loop:**
    - Continuously runs in a loop.
2. **Memory Utilization:**
    - Calls `getMemoryUtilization()` to get the current memory utilization.
    - Prints the result to the console.
3. **CPU Utilization:**
    - Calls `getCpuUtilization()` to get the current CPU utilization.
    - Prints the result to the console.
4. **Pause:**
    - Pauses the execution for 100 seconds using `std::this_thread::sleep_for()`.

Screen Shots
Basic Options and commands



Examples on basic commands



```
Memfs:>create -n file2.txt file3.txt file4.txt
file created successfully
file created successfully
file created successfully

Memfs:>ls -l
File size               created         modified                File Name
0               14/11/2024                              file2.txt
0               14/11/2024                              file3.txt
0               14/11/2024                              file4.txt

Memfs:>write file3.txt Hello_World
11
successfully written to file3.txt

Memfs:>read file3.txt
Hello_World
Memfs:>ls -l
File size               created         modified                File Name
0               14/11/2024                              file2.txt
11              14/11/2024              14/11/2024              file3.txt
0               14/11/2024                              file4.txt

Memfs:>delete -n file2.txt file3.txt file4.txt

Memfs:>ls
Error : There is no files

Memfs:>

Memfs:>exit
----------------Terminal Closed-----------------
```