

OS Project: MemoryOS

Team Members:

Abdullah Irshad(22k-4886)

Amir Abbasi (22k-4923)

Ahmed Ali (22k-4937)

Course Teacher: Engr. Zakir Hussain

Lab Instructor: Engr Afana Malik

Table of Content

|  |  |
| --- | --- |
| Content | Page Number |
| Introduction | 3 |
| Literature Review | 4 |
| Methodology | 5 |
| Costing | 19 |
| Results | 19 |
| Conclusion | 20 |
| Reference | 21 |

Introduction:

Motivation

The Operating system is the backbone of an computing system so that’s why it is very essential and important for a student to understand the concepts of internal mechanism of OS functionalities like the process scheduling, memory management and the file system. By implementing this project the very large gap will be filled between the theoretical concepts and practical implementation. The project also aims on the graphical user Interface to make a good educational environment.

Problem statement with targeted SDGs

The problem addressed is that there is no proper educational tools that allows the student to explore the OS core concepts like the file systems, process management and memory usage.

Targeted SDG : SDG 4 – Quality Education

The projects aims to provide quality education tools to enhance practical knowledge of OS operations.

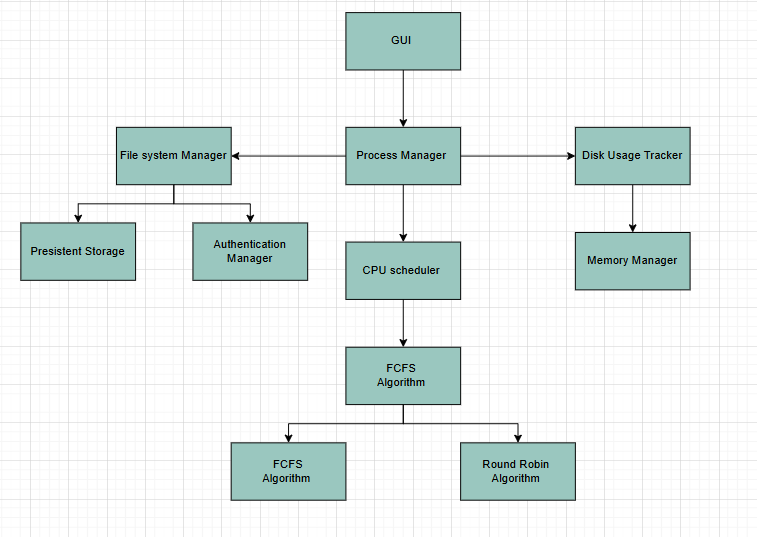
Application to development Board:

The implementation can be later interfaced with the simple Raspberry Pi or ARM development board.

Proposed Solution with Block Diagrams

The solution is that the project solve four main components like the File system manager, the process manager, Disk usage tracker and the CPU scheduler. Each component interacts with the user through a tab based graphical interface.

Flow Diagram made on Draw.io



Literature Review:

1) Avissar proposes about a compiler technique that will automatically partition data across heterogeneous memories in embedded chips, improving the runtime by 39% without programmer effort.

2) Harki discussed about the CPU scheduling techniques in OS focusing on the implementation and performance to improve the management and efficiency.

3)Mazumder discussed that the LoggerFS is Fast lightweight and reliable file system for sensor networks using RAM,FRAM and flash and can achieve up to 800% performance.

4) Norman Discussed about the spectra 70/46 time sharing OS which is designed to combined time sharing and multiprogramming which will support the 48 different controversial users.

5) Kode discusses that reentrant are the fasted synchronized method across Oses, but mutux based mechanism offered more stable performance.

6) Buszta explained how the OS manages and handles the file permission in Window and Linux.

7) John MD discusses that how OS help solve the science puzzle and help understand the multithreading and process management

8) Kaur discussed about a framework that will help user help built light weight applications reducing the OS size and boot time by 50%

9) Chen Discusses about the two real time disk scheduling algorithm which are SSEDO and SSEDV they will outperform the existing method by 38% in minimizing transaction loss.

10) Scott A discusses about the RAD model and RBED scheduler which support the hard real time, soft time and best effort process in complex real time system

Methodology

Created two files FileSystem.java, its purpose is backend operation for file operations, memory management, process management, and scheduling.

The second file is FileSystemGUI.java its purpose is about the frontend graphical user interface using tabs and buttons to interact with the backend.

File and Folder Management

public static TreeMap<String, StringBuilder> files = new TreeMap<>();

public static TreeMap<String, Set<String>> folders = new TreeMap<>();

public static Map<String, String> filePasswords = new HashMap<>();

public static int diskCapacity = 50000;

public static int usedSpace = 0;

Explanation

This parts uses the TreeMaps to store the files and the folders where each file is name maps to its content and the folders map to a set of filenames, the password of the files are saved separately, by counting the disk usage the total disk usage is tracked.

public static String createFile(String filename, String foldername, String content, String password) {

if (files.containsKey(filename)) return "File already exists!";

files.put(filename, new StringBuilder(content));

usedSpace += content.length();

if (foldername != null && !foldername.isEmpty()) {

folders.putIfAbsent(foldername, new HashSet<>());

folders.get(foldername).add(filename);

}

if (password != null && !password.isEmpty()) {

filePasswords.put(filename, password);

}

saveSystem();

return "File \"" + filename + "\" created successfully.";

}

Explanation:

The purpose of this code is to create a new file to decide whether the password protection is required or not, it also updates the memory usage, while also storing the system state to a file.

public static String deleteFile(String filename) { ... }

public static String copyFile(String source, String destination) { ... }

public static String moveFile(String source, String destination) { ... }

Explanation:

This part decides that the files could be copied made and can be copied or moved, when deleting a file the file could free up the memory while copying a file duplicates and the moving option combines the operation of both of them.

public static int getDiskUsagePercent() {

return (int) ((usedSpace \* 100.0) / diskCapacity);

}

Explanation:

Disk Usage Monitoring

public static int getDiskUsagePercent() {

return (int) ((usedSpace \* 100.0) / diskCapacity);

}

Explanation

This part of the code calculates and returns the percentage of used disk space.

Data Persistence (Saving and Loading)

public static void saveSystem() {

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(SAVE\_FILE))) {

oos.writeObject(files);

oos.writeObject(folders);

oos.writeObject(filePasswords);

oos.writeObject(usedSpace);

} catch (IOException e) {

System.out.println("Error saving file system: " + e.getMessage());

}

}

Explanation

This part of the code uses the saveSystem() method which uses the concept of serialization in order to save the state of the file which could be file, folder, password ,disk usage into a binary file for persistence across sessions.

Process Data Structure

static class Process implements Serializable {

String name;

int burstTime;

int arrivalTime;

Process(String name, int burstTime, int arrivalTime) {

this.name = name;

this.burstTime = burstTime;

this.arrivalTime = arrivalTime;

}

}

Explanation

When the process is being created it is stored with its name its burst time which is the execution time and the arrival time into a process object.

Adding a process

public static String addProcess(String name, int burstTime, int arrivalTime) {

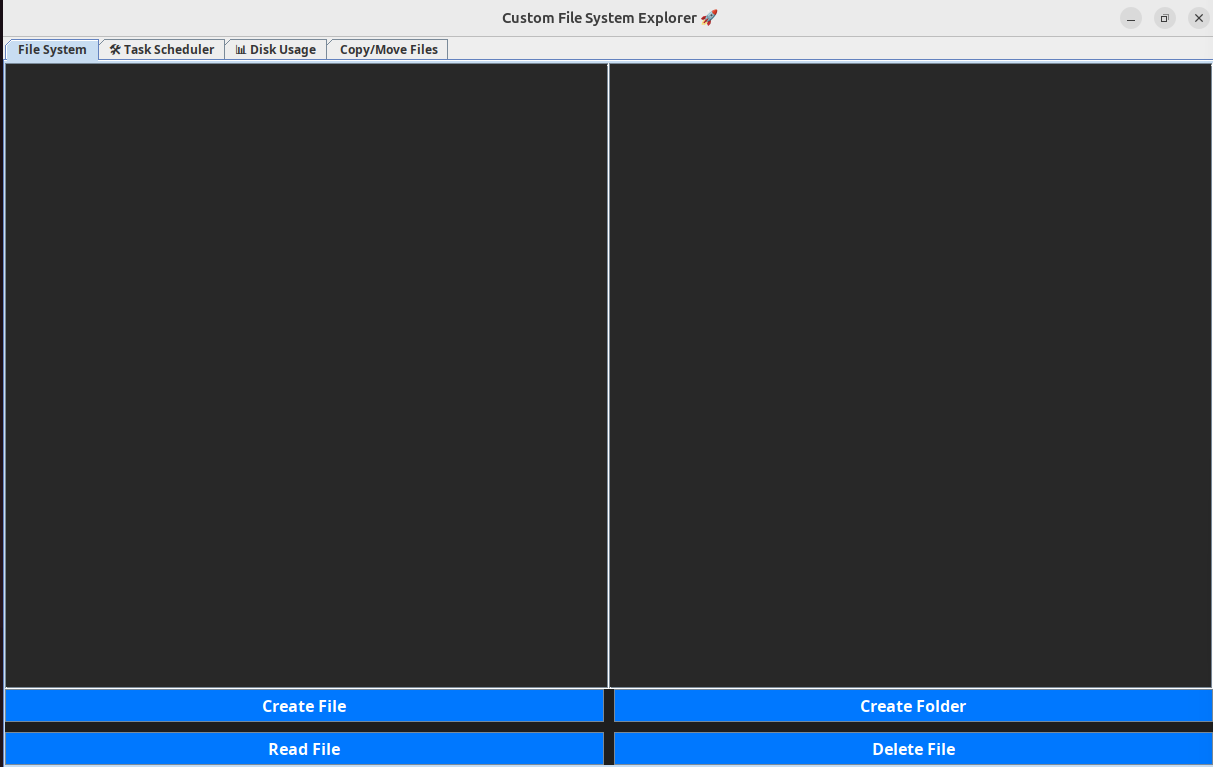
processList.add(new Process(name, burstTime, arrivalTime));

return "Process \"" + name + "\" added.";

}

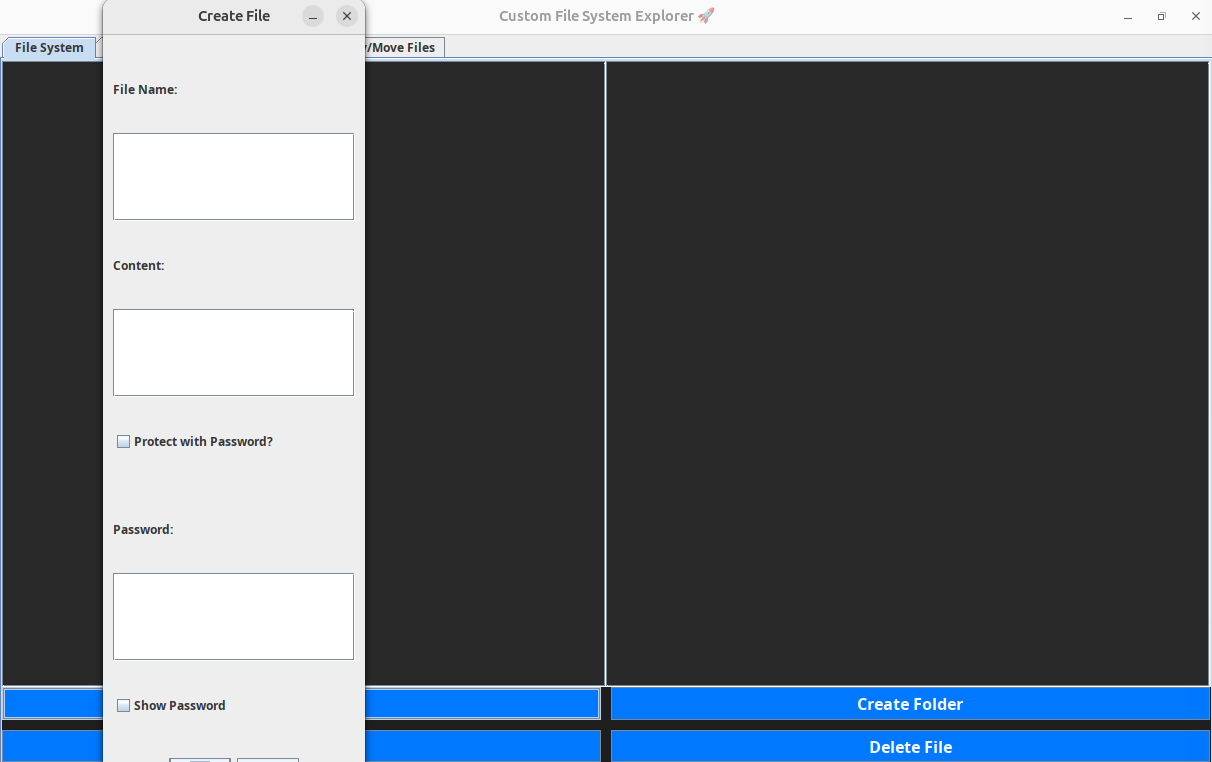
Explanation

This part of the code helps the code allows the user to stored the new processes there which is being stored into the process list for scheduling simulation.

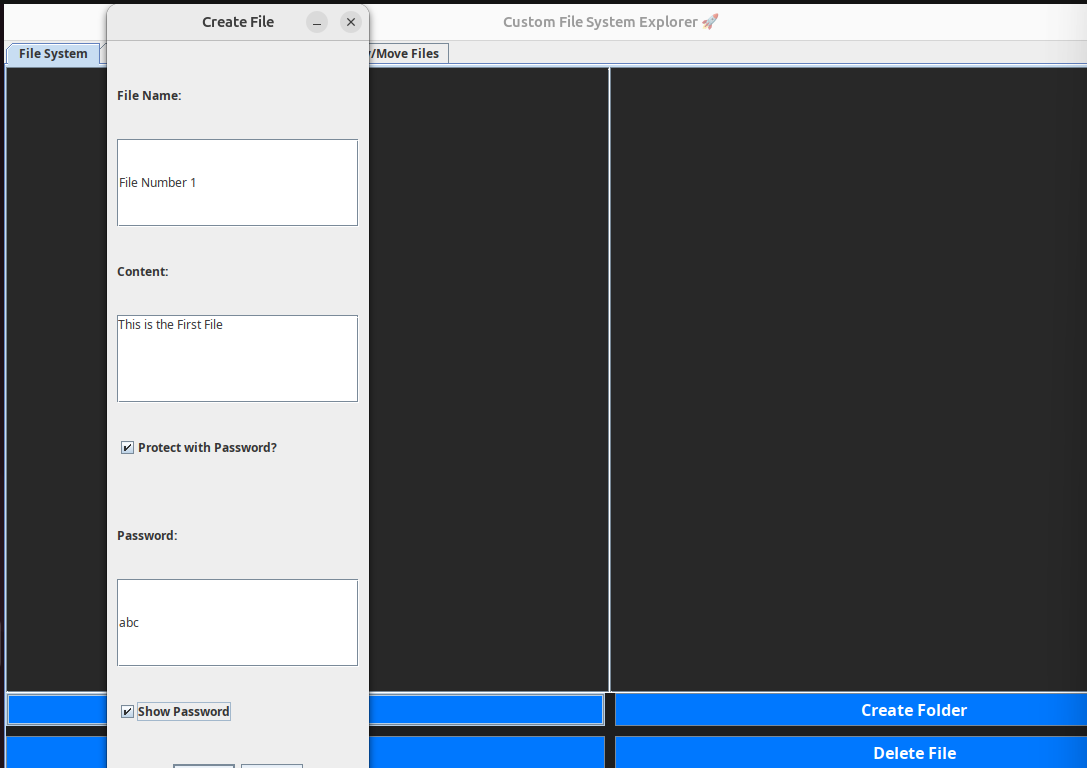


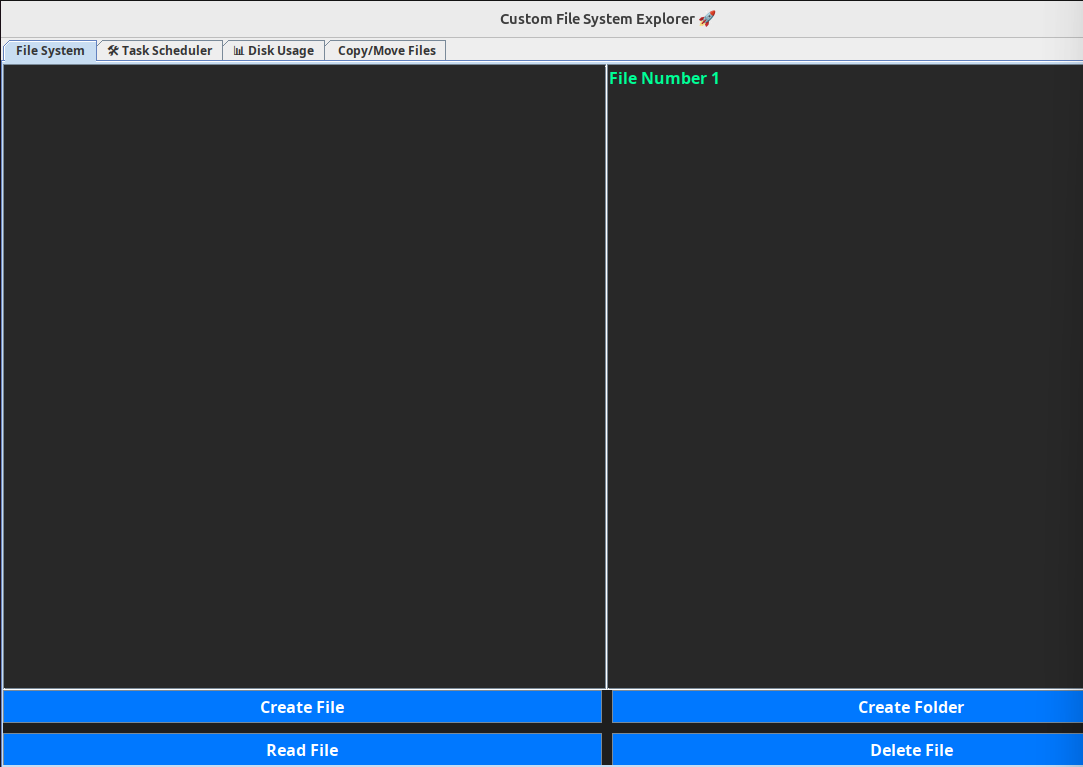
By clicking on the Create File a new process is being created.

Once clicked on the File creation this interface will pop up, It will asked about necessary information like password, the content or description of the file and the File Name, once a password is being setup only the admin can access, read and edit the file.



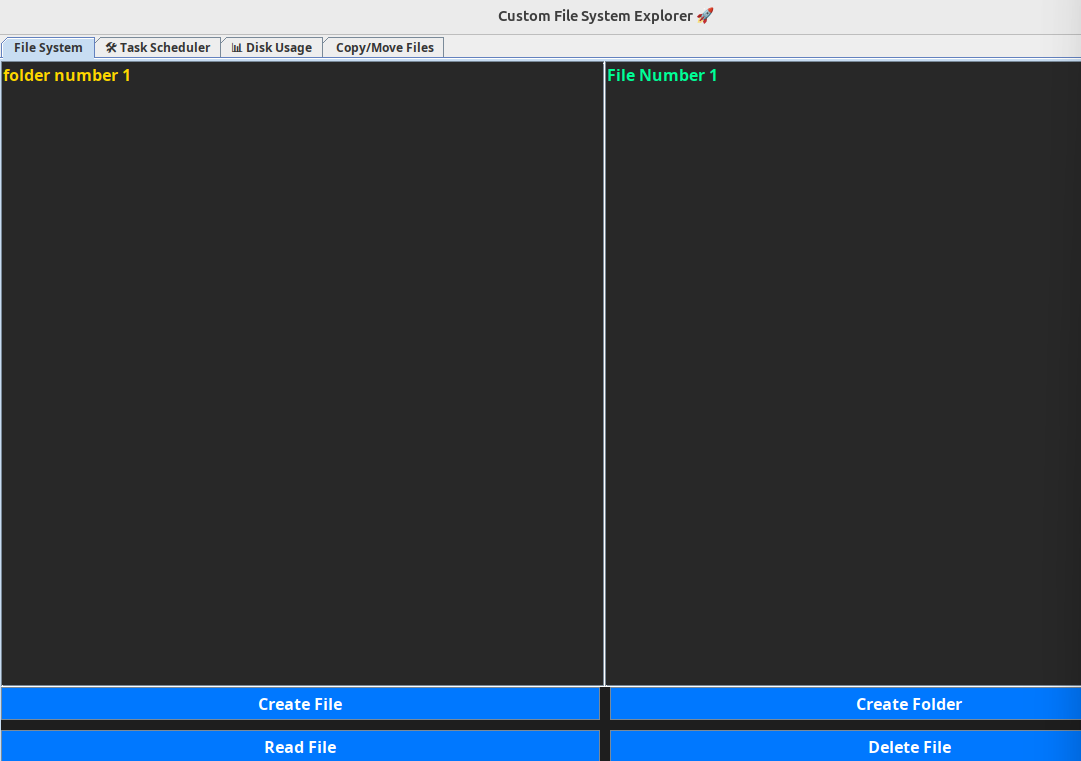
Once given the necessary information click on ok, this will create the process





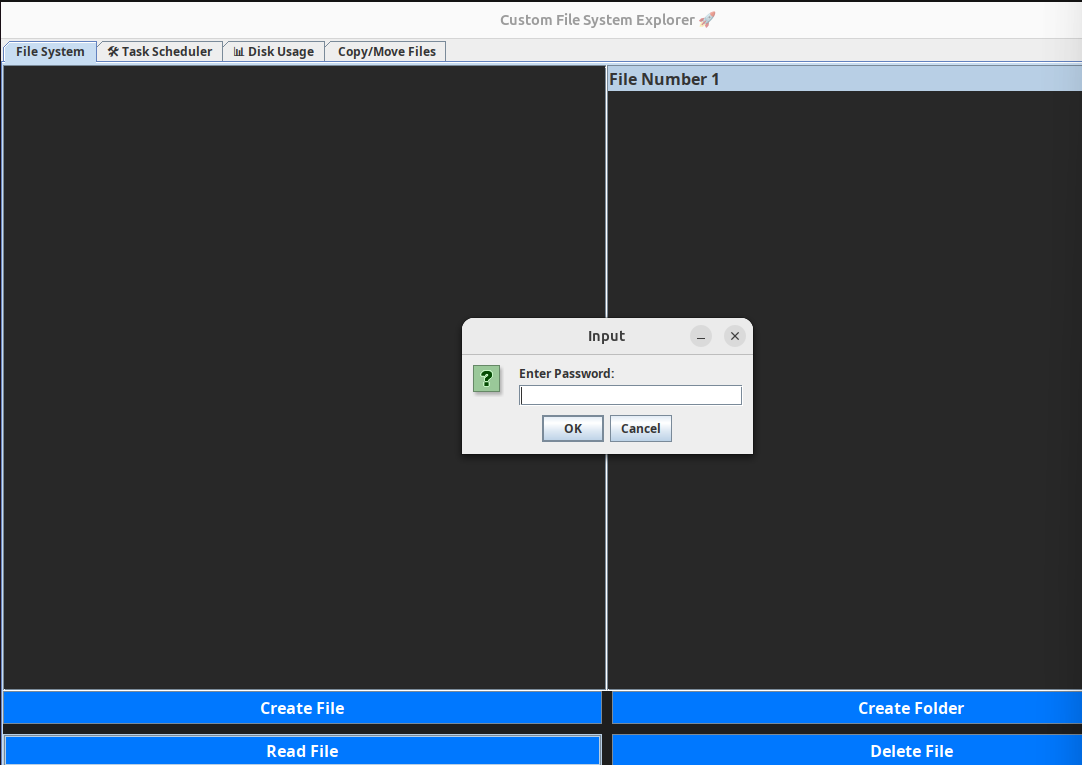
File is being created and is shown in the interface.

The same the folder can also be created just by clicking on the “create folder” on the right side.

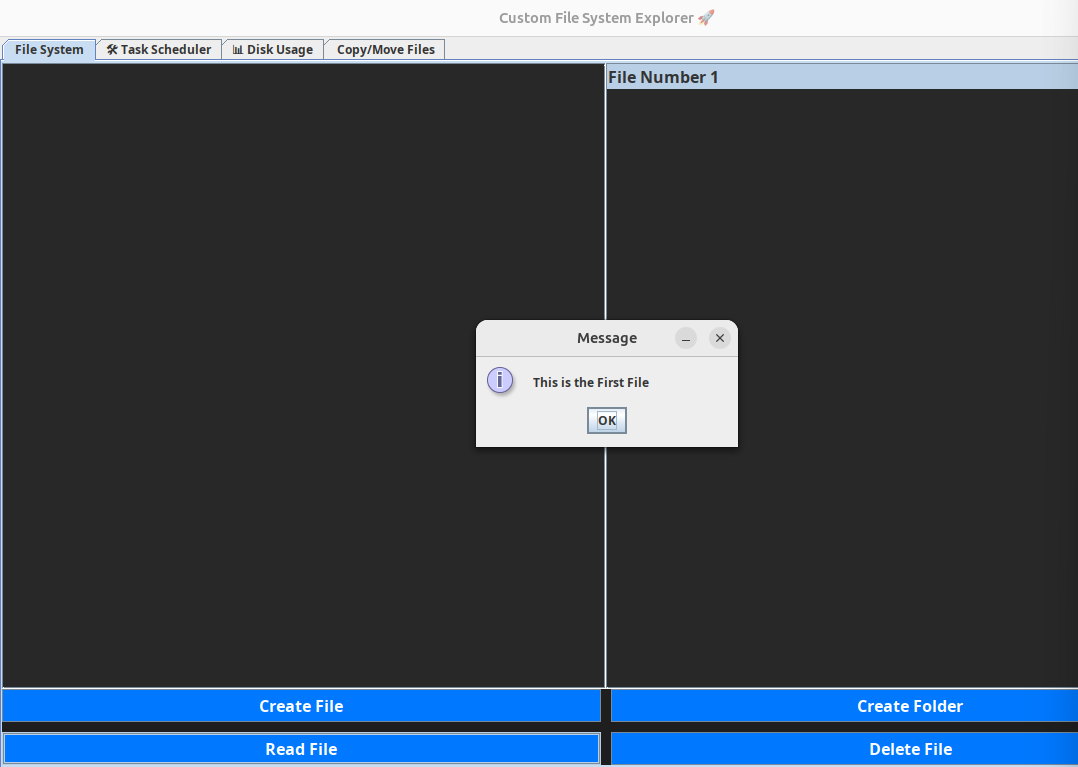


The file will appear on the right side and folders on the left side

Now to read the file the required file will be selected first then click on the Read File Button to read the content of the file, if the password is required to be set then it will first ask the password.



Once given the password you can now read the content of the file. The description of the file is shown in the red box.



Now to delete a file the user will click the file which they want to delete and click on the Delete file button.

First come First Serve (FCFS) Scheduling

public static List<String> simulateFCFSTimeline() {

List<String> timeline = new ArrayList<>();

processList.sort(Comparator.comparingInt(p -> p.arrivalTime));

int currentTime = 0;

for (Process p : processList) {

if (currentTime < p.arrivalTime) currentTime = p.arrivalTime;

timeline.add(p.name);

currentTime += p.burstTime;

}

return timeline; }

The FCFS scheduling algorithm evaluate and sorts all the processes based on the arrival time, It then schedules them one after the another they all complete the process one after another without the preemption. Also a timeline list is generated which will later help the code to built the Gantt chart

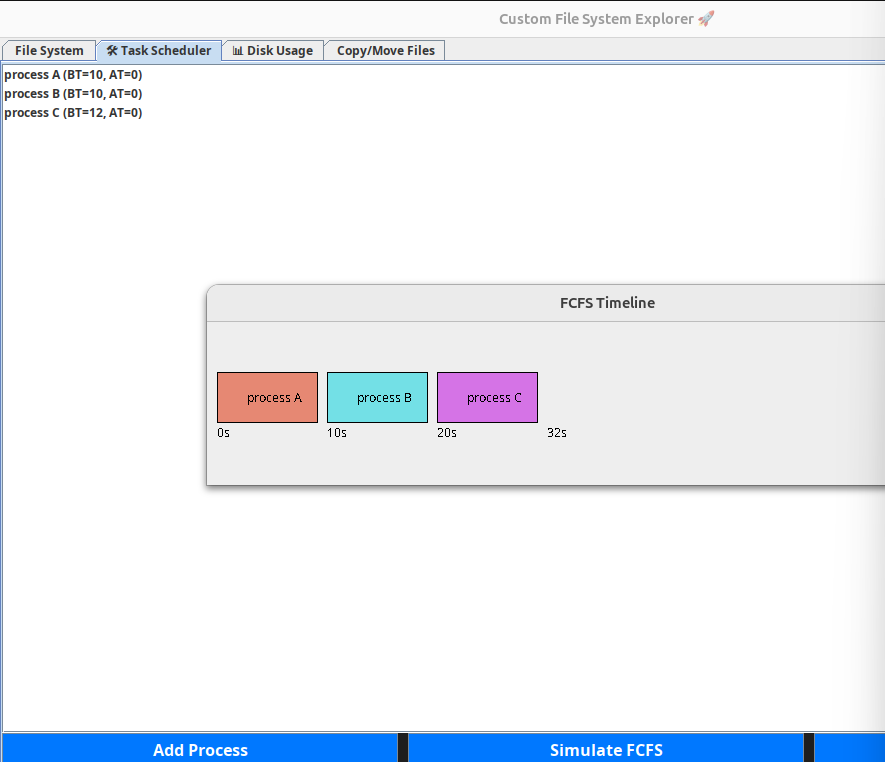
Example for FCFS of 3 processes.

Process A, Arrival time = 0, Burst time = 10

Process B, Arrival time = 0, Burst time = 10

Process C, Arrival time = 0, Burst time = 12

The Gantt chart is also made with the FCFS evaluation to make it more eye catching.



Round Robin scheduling

public static List<String> simulateRoundRobinTimeline(int quantum) {

List<String> timeline = new ArrayList<>();

List<Process> queue = new ArrayList<>();

for (Process p : processList) {

queue.add(new Process(p.name, p.burstTime, p.arrivalTime));

}

int currentTime = 0;

while (!queue.isEmpty()) {

Process p = queue.remove(0);

if (p.burstTime > quantum) {

timeline.add(p.name);

p.burstTime -= quantum;

currentTime += quantum;

queue.add(p);

} else {

timeline.add(p.name);

currentTime += p.burstTime;

}

}

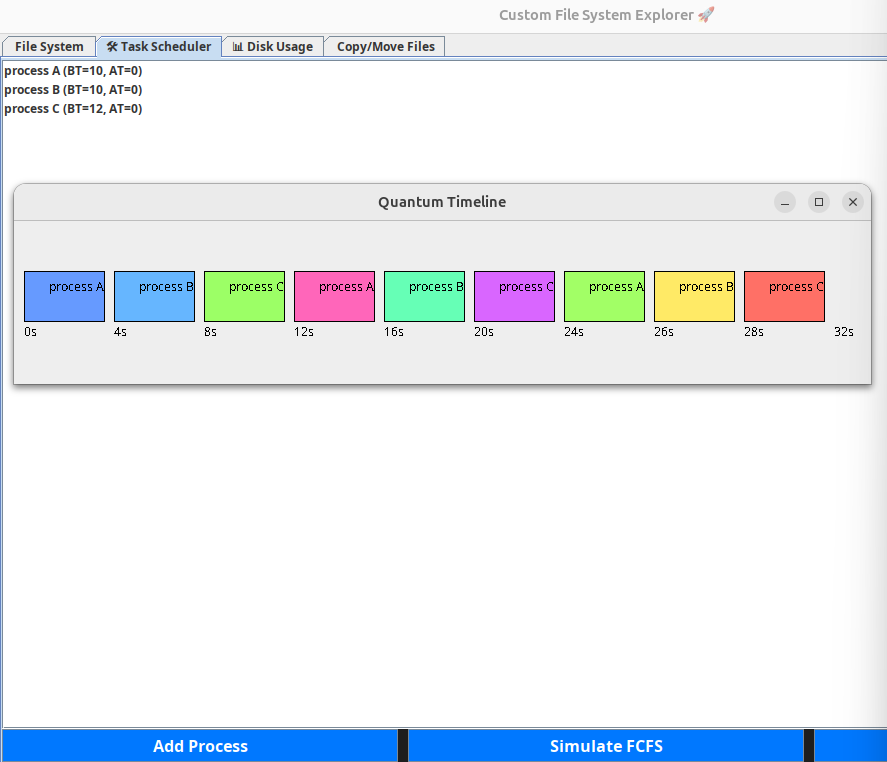
return timeline;

}

Explanation

This part of the code explain about the round robin scheduling the round robin complete its process is a circle wise scenario, one will take on turn to complete its process and then passes to the next one if on the process is not completed in the next turn then it will be processed or meaning that it will take its turn again, if one process is completed then it will skip its turn and passes to the next one, it also take quantum time which is fixed, it is placed back in the queue, the timeline order is stored for the visualization.

Gantt Chart for the Round Robin for the same problem as of the FCFS.



Main GUI layout

JTabbedPane tabs = new JTabbedPane();

tabs.addTab("📁 File System", createFileSystemTab());

tabs.addTab("🛠️ Task Scheduler", createTaskSchedulerTab());

tabs.addTab("📊 Disk Usage", createDiskUsageTab());

tabs.addTab("📂 Copy/Move Files", createCopyMoveTab());

add(tabs);

Explanation

The GUI uses the tabbed pane which will help the code to separate the different functionalities like the file management, process scheduling, disk monitoring and file copying and moving.

File System Management Tab

private JPanel createFileSystemTab() { ... }

Explanation

This part of the code provides and interface for creating, reading, deleting files and folders.

It also protected the password supported file using boxes and password verification.

Task Scheduler Tab

private JPanel createTaskSchedulerTab() { ... }

Explanation

This part of the code helps the users to add new processes also taking the burst time and arrival time as the input and simulate scheduling using the FCFS and Round Robin.

Disk Usage Monitoring Tab

private JPanel createDiskUsageTab() { ... }

Explanation

This part will tell you about the disk usage percentage dynamically through a process bar based on the number and the size of the file which is being created.

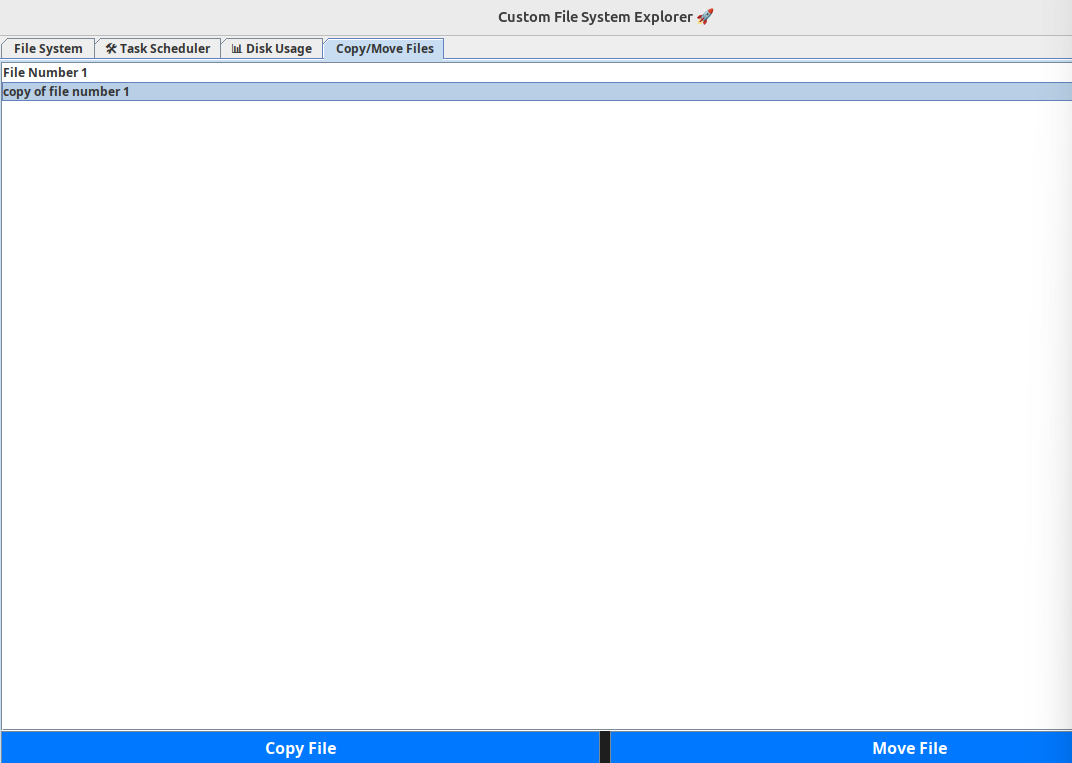
Copy/Move Files Tab

private JPanel createCopyMoveTab() { ... }

Explanation

This part of the code will allow the users to copy the file also delete the file and also help in moving the file from one destination to another.

Now for the GUI of move/copy file the interface is below, to copy a file first select the file click on copy.



Same way we can also move a file

FCFS and Round Robin Gantt Chart Visualization

private class QuantumDiagram extends JPanel { ... }

private class FCFSDiagram extends JPanel { ... }

Explanation

This part of the code is for the graphical representation of the Gantt Charts, the dynamic panels the Gantt chart based on the simulation timelines, displaying each process block diagram with random colors and time markings for easy understanding.

Costing

Java Development Kit = Free

Java Swing GUI Toolkit = Free

Development Time = 50 hours (Group Contribution)

Draw.io = Free

Results

This project allows the user to create file, folder, delete file and folder, rename them and also make a copy and move them, this all is achieved using simple GUI interface, the files could also be password protected and user must enter the correct password to access secured files

This project uses the disk management, as we create more and more files data is being increased and with that the disk storage is also being increased this gives a real time representation of memory utilization.

This project also helps us calculate the scheduling algorithm such as FCFS and Round Robin, the FCFs will take the input like its process name, burst time, arrival time and for the Round Robin the Quantum is also given as input

Both the FCFS and Round Robin generate the Gantt using the GUI interface small different colors represents the color and the Gantt chart is accurate with the required processes being stated. This will help the user to easily understand how scheduling algorithm works in real practice.

The File systems structure, files, folder, password and disk usage are saved automatically using the concept of serialization, whenever a user closes the program and they login again then all the previous data before closing the program is restored again correctly, which highlights the continuity of the system.

Conclusion

The memory operating system successfully implemented the functions like the file management system, disk usage monitoring, and the process scheduling which uses the FCFS and Round Robin algorithm. The Graphical user interface provides a simple and user friendly environment for learning experience and it also helps the user to interact with the operating system, additionally the Gantt chart successfully generate accurately the process, making the complex and hard scheduling easier to understand. Overall, the project fulfills the requirement which was being stated. In the Future, it can be further improved by adding more advanced scheduling algorithms also we can further add file compression or we can also introduce real time memory monitoring partitioning features to make the simulation even closer to the real world operating system.

Reference:

Avissar, Oren, Rajeev Barua, and Dave Stewart. "Heterogeneous memory management for embedded systems." Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems. 2001.

Harki, Naji, Abdulraheem Ahmed, and Lailan Haji. "CPU scheduling techniques: A review on novel approaches strategy and performance assessment." Journal of Applied Science and Technology Trends 1.1 (2020): 48-55.

Mazumder, Biswajit, and Jason O. Hallstrom. "A fast, lightweight, and reliable file system for wireless sensor networks." Proceedings of the 13th International Conference on Embedded Software. 2016.

Weizer, Norman, and G. Oppenheimer. "Virtual memory management in a paging environment." Proceedings of the May 14-16, 1969, spring joint computer conference. 1969.

Kode, Oluwatoyin, and Temitope Oyemade. "Analysis of Synchronization Mechanisms in Operating Systems." arXiv preprint arXiv:2409.11271 (2024).

Buszta, Ken. "Directory Security." Information Security Management Handbook. Auerbach Publications, 2019. 747-758.

Hill, John MD, et al. "Puzzles and games: addressing different learning styles in teaching operating systems concepts." Proceedings of the 34th SIGCSE technical symposium on Computer science education. 2003.

Kaur, Jasleen, and S. R. N. Reddy. "Design and development of lightweight operating system framework for smart devices." International Journal of Software Innovation (IJSI) 9.2 (2021): 136-149.

Chen, Shenze, et al. "Performance evaluation of two new disk scheduling algorithms for real-time systems." Real-Time Systems 3.3 (1991): 307-336.

Brandt, Scott A., et al. "Dynamic integrated scheduling of hard real-time, soft real-time, and non-real-time processes." RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003. IEEE, 2003.