Project Report: Multi-Process/Threads Manager with IPC and Parallel Text File Processing

Description:

This project simulates an operating system's process and thread management, inter-process communication (IPC), and text file processing. It aims to provide practical experience in comprehending and handling multiple processes and threads, showing how they interact and share assets within a system. The project also examines IPC mechanisms, providing individuals a hands-on opportunity to learn how processes can interact efficiently in a controlled environment.

In addition to process and thread management, the project examines parallel computing by creating a system to process text files. It is an in-depth guide to understand the complex topics of system administration, IPC, and the benefits of computing in parallel in real-world applications.

Structure of the Code

ProcessInfo and ThreadInfo structures are used to manage and store data about processes and threads.

SharedMemory Structure: enables for IPC through shared memory.

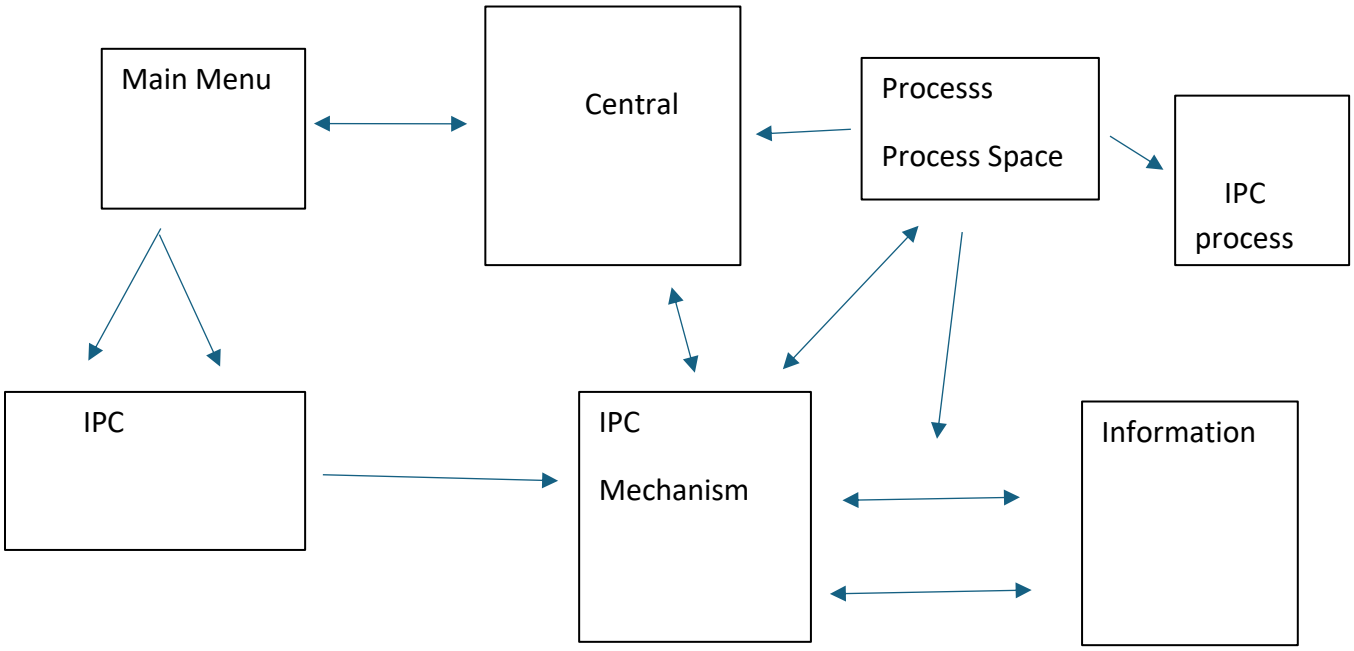
display Function: Displays the current state of processes

process_text_file Function: Operations text files by converting characters to uppercase and counting their occurrences.

simulate_ipc_processes Function: Displays IPC with shared memory.

main Function: Showing a menu to the user while accepting input to manage handles and threads, simulate IPC, or process text files.

Diagram

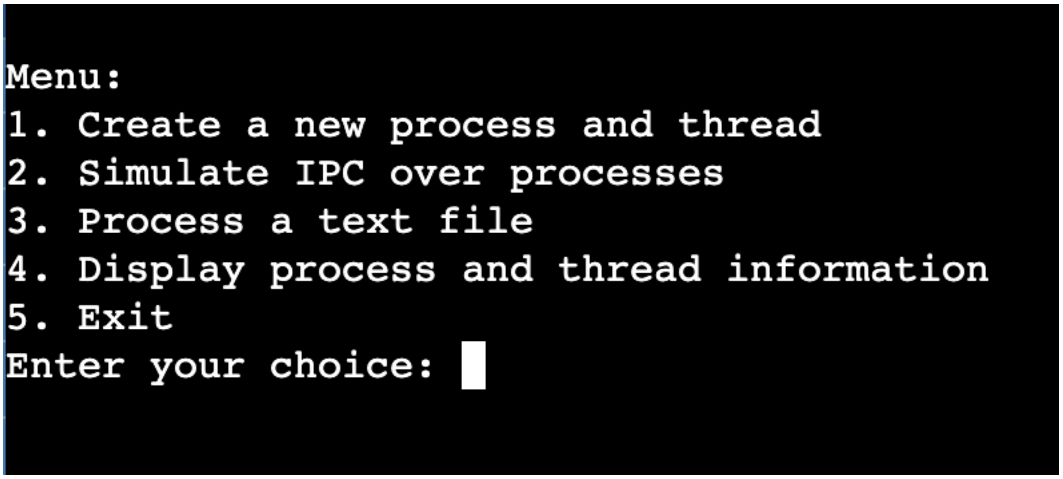


Code Implementation

Starting the Program:

When the program is run, it displays the main menu with options 1 to 5.

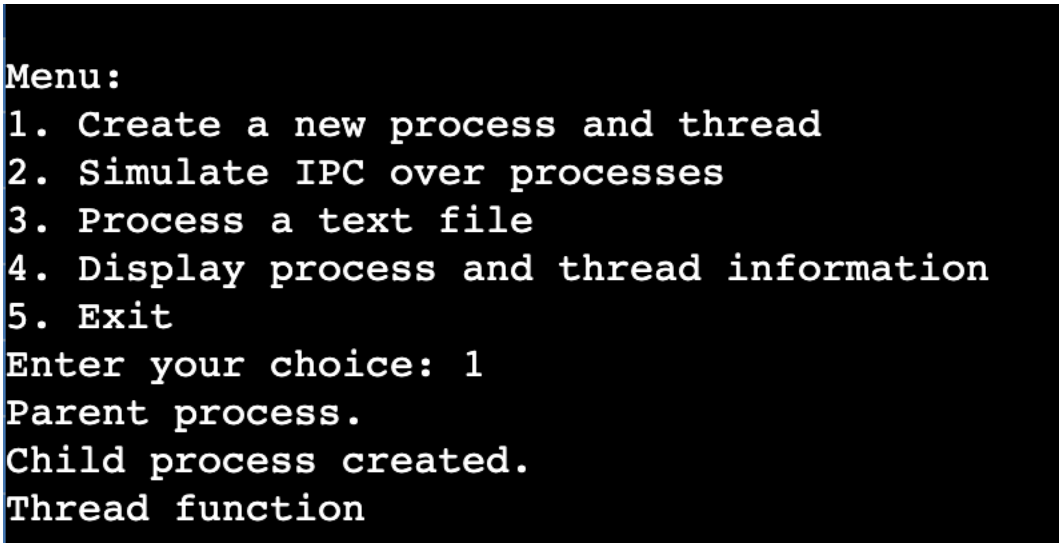
Menu Option 1: Create a New Process and Thread



Expected Console Output:

Child process: Child process created. and Thread function"from the thread.

Parent process: "Parent process."



Menu Option 2: Simulate IPC over Processes

The program creates a shared memory segment and writes a message to it.

Expected Console Output: "Message from shared memory: Hello from shared memory"

```
Menu:
1. Create a new process and thread
2. Simulate IPC over processes
3. Process a text file
4. Display process and thread information
5. Exit
Enter your choice: 2
Message from shared memory: Hello from shared memory
```

Menu Option 3: Process a Text File

The program processes a text file named Example.txt and prints the uppercase version of alphabetic characters and their counts.

Expected Console Output: Character counts, e.g., D:1, E:1, L:3

```
Menu:
1. Create a new process and thread
2. Simulate IPC over processes
3. Process a text file
4. Display process and thread information
5. Exit
Enter your choice: 3
D: 1
E: 1
H: 1
L: 3
O: 2
R: 1
W: 1
```

Menu Option 4: Display Process and Thread Information

The program displays the status of all processes and their threads created so far.

Expected Console Output: A table showing process IDs, their status, the number of threads, thread IDs, and their status.

```
Menu:
1. Create a new process and thread
2. Simulate IPC over processes
3. Process a text file
4. Display process and thread information
5. Exit
Enter your choice: 4
Process ID      Status  Num Threads
1333            Running 1
Thread ID       Status
140162229278272 Running
```

Menu Option 5: Exit

The program exits.

Expected Console Output: "Exiting program."

```
Menu:
1. Create a new process and thread
2. Simulate IPC over processes
3. Process a text file
4. Display process and thread information
5. Exit
Enter your choice: 5
Exiting program.
```

Code Sanity Check

Process and Thread Management: The processes and threads are created as expected and their information is correctly displayed, then this part of the program is working properly.

IPC Simulation: The generation of shared memory divides and message passing show that the IPC mechanisms are in operations.

Text File Processing: The application properly reads "Example.txt", converts the letters, and counts them, it means that the file processing part is operational.

Project Findings and Challenges

Keeping thread safety, particularly in shared data scenarios, was difficult.

Managing and tracking the resource use of processes and threads required careful consideration.

As the number of processes and threads grows, maintaining performance and managing resources become more and more difficult.

Limitations and chances for improvement

Currently, the project has a basic console interface. A graphical interface could improve user experience.

The current implementation is basic. More advanced IPC methods could provide greater understanding.

More error handling and logging mechanisms could be implemented to improve debugging and user feedback.

Conclusions

This project successfully demonstrated a thorough simulation of process and thread management, inter-process communication (IPC), and parallel text file processing in a simulated operating system. The implementation shows fundamental system programming ideas such as process creation and management, thread handling, and shared memory IPCThe addition of text file processing shows the benefits of computing in parallel for real-world data processing tasks.

Throughout the project, various challenges were encountered and overcome, such as concurrency issues in thread management, IPC complexities, and the specifics of efficient text file processing. The project demonstrates the importance of understanding behind system processes and how they interact, which is important for a CS student.