Brianna Zaffina

Professor Han

CPSC 351-01

22 June 2025

Design of Sender and Receiver

Introduction:

This project creates an interprocess communication (IPC) system using shared memory and message queues. Using this, the Sender is able to take the information from a specified file with a message inside of it, count the bites, and send it to the Receiver which then takes this message, counts the bites, and puts it into a received file.

Design Levels:

1. **Architectural Design:**

   The two main components of this project consist of the **Sender process** and the **Receiver process.** They communicate to each other using shared memory and message queues which then allows us to use asynchronous communication that retrieves messages from a process in an orderly manner (GeeksForGeeks).

2. **High-Level Design:**

   Sender Module:

   - Takes the original file and retrieves message
   - Writes message (data) to shared memory
   - Sends message from file to the Reciver using message queue

   Receiver Module:

   - Receives message from Sender using message queue

- Reads shared memory
- Creates new receiver file with message received from Sender

Signal Handle Module:

- Frees system V resources (Ctrl-C)

Error Handeling Module:

- Ensures system call values and file handling values are usable
- Uses perror() to output any potential errors
- Terminates and cleans up resources
- Ensures no memory leaks

3. **Detailed Design:**

   Each module has its functions that implement the IPC:

   recvFileName()/sendFileName(): Manages communication

   sendFile(): Sends file to receiver

   ctrlCSignal(): Creates signal to free resources

   cleanUp(): Detaches shared memory and deallocates message queue

   init(): initializes key, memory segment, and message queue

Concurrency:

The Sender and Receiver are able to run concurrently alongside each other, and it is actually necessary for this. The Receiver must be open and ready to take any information from the Sender. The Sender must receive the file name and start the process of sending the information to the Receiver.

Cohesion:

Sender and Receiver are classified as Communicational Cohesion as they are sequentially executed and work on the same data.

Coupling:

Sender and Receiver are classified as Data Coupling as they interact with each other through the means of sending and receiving data.

Design Verification:

Verified code structure by using references and implementing error handling, ensuring that all the code is valid and would not cause any leaks or problems. Output goes according to project requirements, and the file message is sent from the sender to the receiver effortlessly without any issues. Bytes are counted and received, and a new file is created with the received message.

Conclusion:

This project can be separated into four modules, which are the Sender, Receiver, Signal Handle, and Error Handler. The Sender and Receiver have the ability to concurrently run alongside each other and, as such, are classified as Communicational Cohesion. The Sender and Receiver interact with each other to share a file message, thus classifying them as Data Coupling. Overall, the design of the project is verified.

Works Cited

"IPC Using Message Queues." *GeeksforGeeks*, GeeksforGeeks, 29 May 2025, www.geeksforgeeks.org/ipc-using-message-queues/.

"Software Analysis and Design Tools." *Tutorialspoint*, www.tutorialspoint.com/software_engineering/software_analysis_design_tools.htm. Accessed 21 June 2025.

"Software Design Basics." *Tutorialspoint*, www.tutorialspoint.com/software_engineering/software_design_basics.htm. Accessed 21 June 2025.