Design of Sender and Receiver

Authors
Christian Verry
Eddie Poulson
Thanedh Sithisombath
Greg Zhang

Functional Description

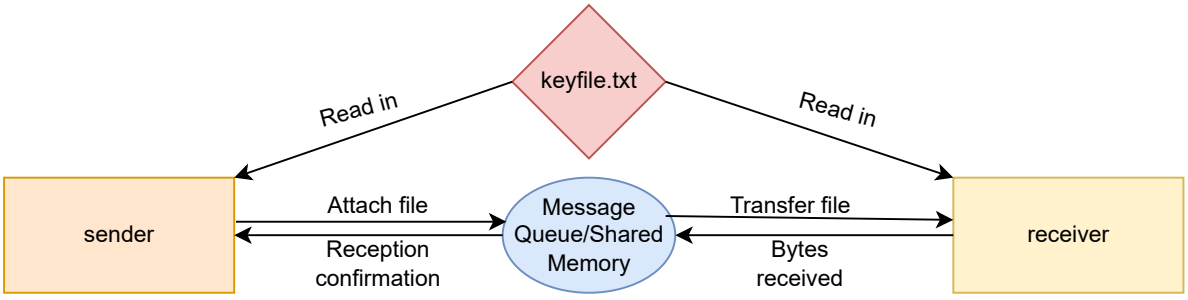
Application which applies shared memory and message queues to implement file transfers between two separate systems. Split between two programs: sender and receiver.

sender.cpp

1. init()
    - Generate key using "keyfile.txt"
    - Attach to shared memory and message queue
2. cleanup()
    - Detach pointer from the shared memory
3. sendFile()
    - Open and read the entire file
    - Count the size of the message sent to the receiver
    - Wait on receiver to confirm
4. sendFileName()
    - Use "msgsnd" to send file name and size of name
5. main()
    - Shared memory and message queue: init(shmid, msqid, sharedMemPtr);
    - Send the name of file: sendFileName(argv[1]);
    - Send the file: fprintf(stderr, "The number of bytes sent is %lu\n", sendFile(argv[1]));
    - Cleanup process: cleanUp(shmid, msqid, sharedMemPtr);

recv.cpp

1. recvFileName()
    - Get the file name from the sender program using "mgrcv"
2. init()
    - Use "ftok" to generate a key from "keyfile.txt"
    - Allocate and attach shared memory, create message queue
3. mainLoop()
    - Open file for transfer using "fopen"
    - Loop until "msgSize" = 0 then use "fclose" to close file once completed.
4. cleanUp()
    - Detach and deallocate shared memory and message queue
5. ctrlCSignal()
    - Free resources by calling cleanUp function
6. main()
    - Install a signal handler: signal(SIGINT, ctrlCSignal);
    - Set up shared memory and queue: init(shmid, msqid, sharedMemPtr);
    - Receive file name from sender: string fileName = `recvFileName();`
    - `Go to mainLoop:` fprintf(stderr, "The number of bytes received is: %lu\n", mainLoop(fileName.c_str()));
    - Call cleanup: cleanUp(shmid, msqid, sharedMemPtr);



Goals and Milestones

- Implement and understand IPC principals
- Utilize shared memory and messaging queues for efficient transfer of data between multiple programs (Sender & Receiver).
- Sender sends all information in file to receiver and signals end.
- Receiver receives information until it receives a field size of 0 bytes from the sender program.
- Both sender and receiver programs are able to deallocate shared memory and message queues.