1. Data Flow:

RecvFile (output)

Keyfile (input)



Shared Memory

1. Pseudo code:

Sender:

Init:

Generate shared memory key

Allocate part of shared memory

Attach to shared memory using a pointer (set up by receiver)

Make a message queue

Send:

While (not at end of file)

Read number of bytes and store in shared memory

Send message telling receiver data is ready

Wait for receiver to send message that it saved the memory chunk

Send message to receiver with size field set to 0 (indicating there are no more messages to send)

Cleanup:

Close file

Detach from shared memory

Exit

receiver:

signal:

Install signal handler

init:

Generate key for shared memory

Allocate a piece of shared memory

Attach to shared memory with a pointer

Make a message queue

mainloop:

Open file for writing

Receive message and determine message size

While (message size =/= 0)

Read number of bytes from shared memory and save to file.

Send message to sender confirming reception and saving of data

cleanup:

Close file

Detach from shared memory

Exit

1. Flowchart Design

Sender Program Flow Chart

sender.cpp

Init

Initialize, create key, attach to shared memory and message queue

void init(int& shmid, int& msqid, void\*& sharedMemPtr)

↓

Send

Open file for reading, store data in shared memory, notify receiver and wait for reception confirmation, then message receiver that nothing more to send, close file

void send(const char\* fileName)

↓

Cleanup

Detach from shared memory

void cleanUp(const int& shmid, const int& msqid, void\* sharedMemPtr)

Receiver Program flow chart

recv.cpp

|  |
| --- |
| Signal  IPC  signal.h |

 

|  |
| --- |
| Initialize  Create file, generate key, allocate shared memory  void init(int& shmid, int& msqid, void\*& sharedMemPtr) |



|  |
| --- |
| Main loop  Gets message & size, open file, checks,  Void mainloop() |

|  |
| --- |
| Else  exit |

  
 

|  |
| --- |
| Detach & deallocate  void cleanup(const int& shmid, const int& msqid, void\* sharedMemPtr) |