Design of Assignment 1

Introduction

The big idea is to design a “Sender” and a “Receiver”, and the job of the “Sender” is to break down a large amount of message to little chunks, and send it to “Receiver”, and only one chunk of message could be “on the way” at a time. Also, the communication should be both directions, which means both “Sender” and “Receiver” can send and receive message.

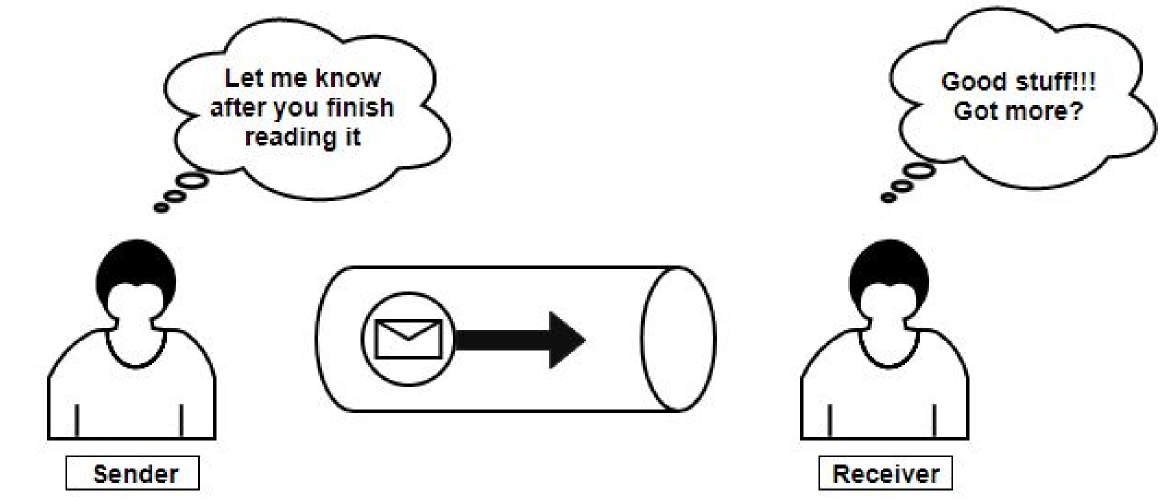

Figure 1.0

Overview of this project

Goal in this project is designing two programs that communicate to each other by using "Message queue" and "Shared Memory" technique. Therefore, we need to add shared memory in addition to *Figure 1.0*.

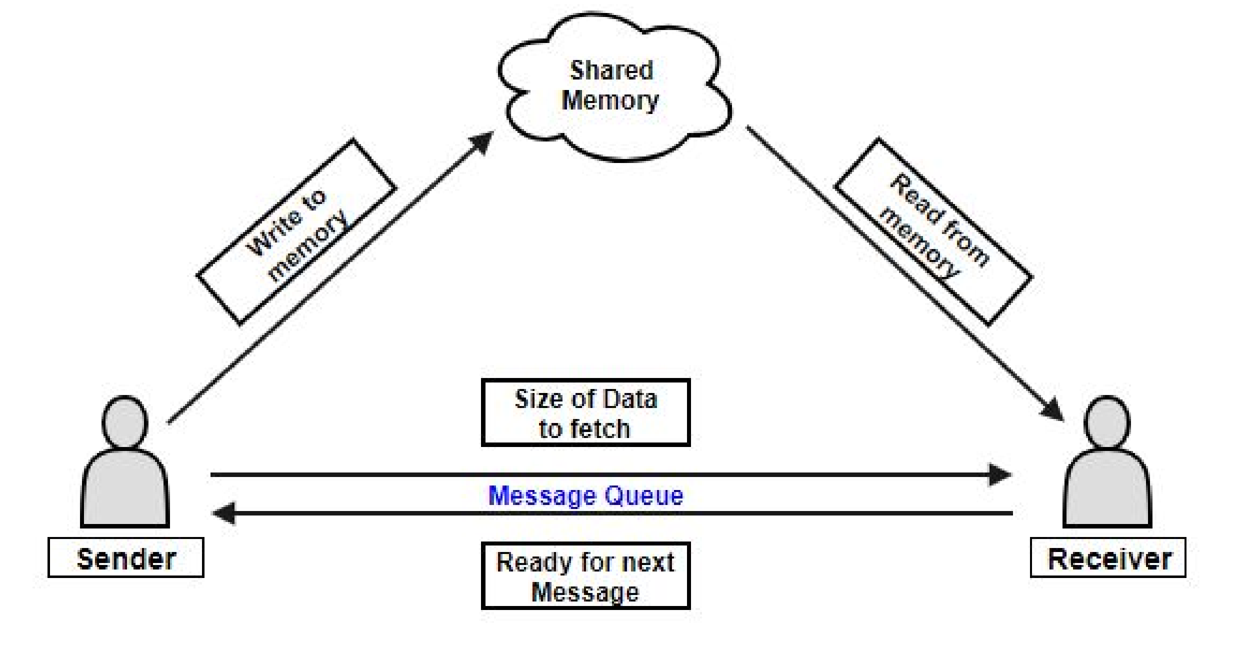

Figure 1.1

This is the overview of this project, we added "Message Queue" and "Shared Memory", the purpose of Message Queue is to let Receiver knows the size of message need to be fetched from Shared Memory. After Receiver got the message from the Shared Memory, the Receiver will send a message to Sender, let Sender know about this and wait for next Message. After Sender got the signal from Receiver, Sender will store message into Shared Memory and then will send the size of the message to Receiver again until there is no message anymore, Sender will send 0 to indicate that is the end of the communication.

More Details

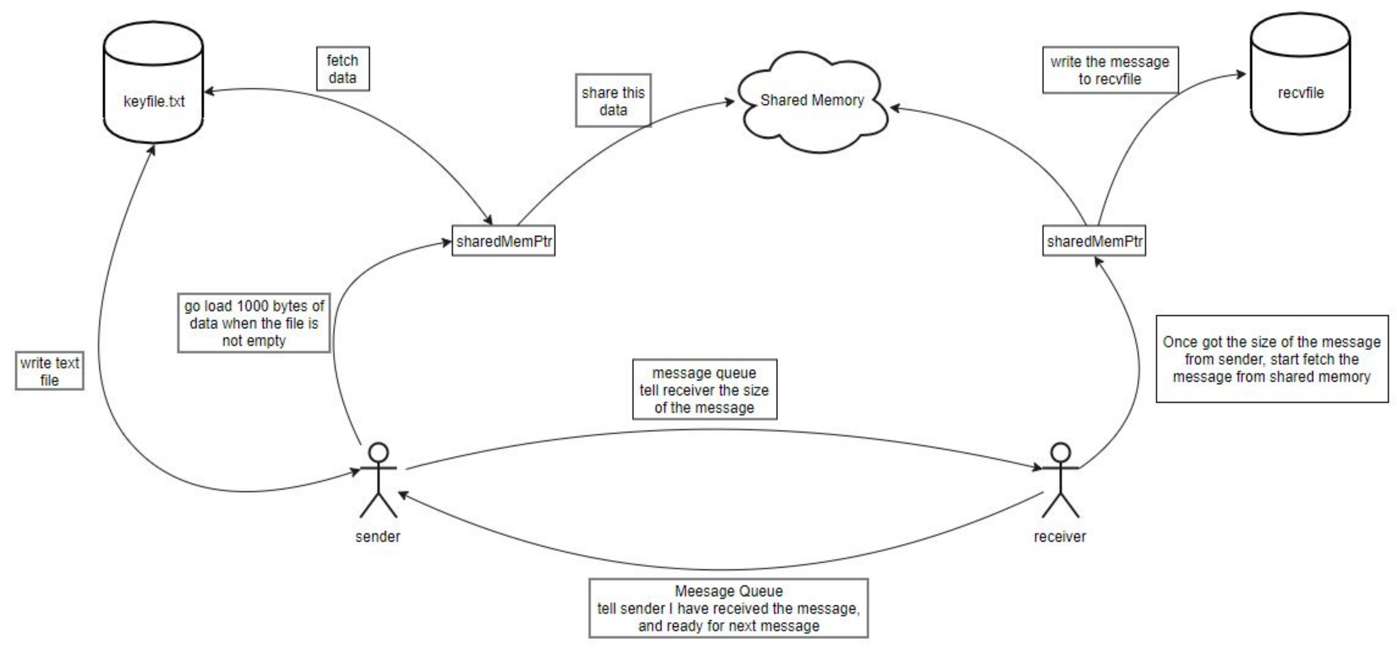
keyfile.txt
fetch data
share this data
Shared Memory
write the message to recvfile
recvfile
sharedMemPtr
sharedMemPtr
go load 1000 bytes of data when the file is not empty
write text file
message queue tell receiver the size of the message
Once got the size of the message from sender, start fetch the message from shared memory
sender
receiver
Meesage Queue tell sender I have received the message, and ready for next message

Figure 1.2

Figure 1.2 - with more details added, disk and pointers got involved.

Extra Credit Design

Notice we have **Extra Credit**, so we made a design for **Extra Credit**, the idea of extra credit is to replace Message Queue with Signal, but we notice Signal does not have the ability to send message, it sends only signal instead.

So we came up with an idea that store 4 bytes int size into the Shared Memory, let Receiver fetch the size first, and then store the real message into Shared Memory. Each time Sender store data to Shared Memory, the sender will also signal Receiver to read from the memory, and each time Receiver done with fetch information from Shared Memory, the Receiver will notice Sender.

Prerequisite:
Create a ***boolean*** variable called ***"size_mesg"***, and initialize it to true, in both Sender and Receiver, and each time they fetch or write information to Shared Memory successful, they need to do "***not" operation*** to the ***"size_mesg"***.

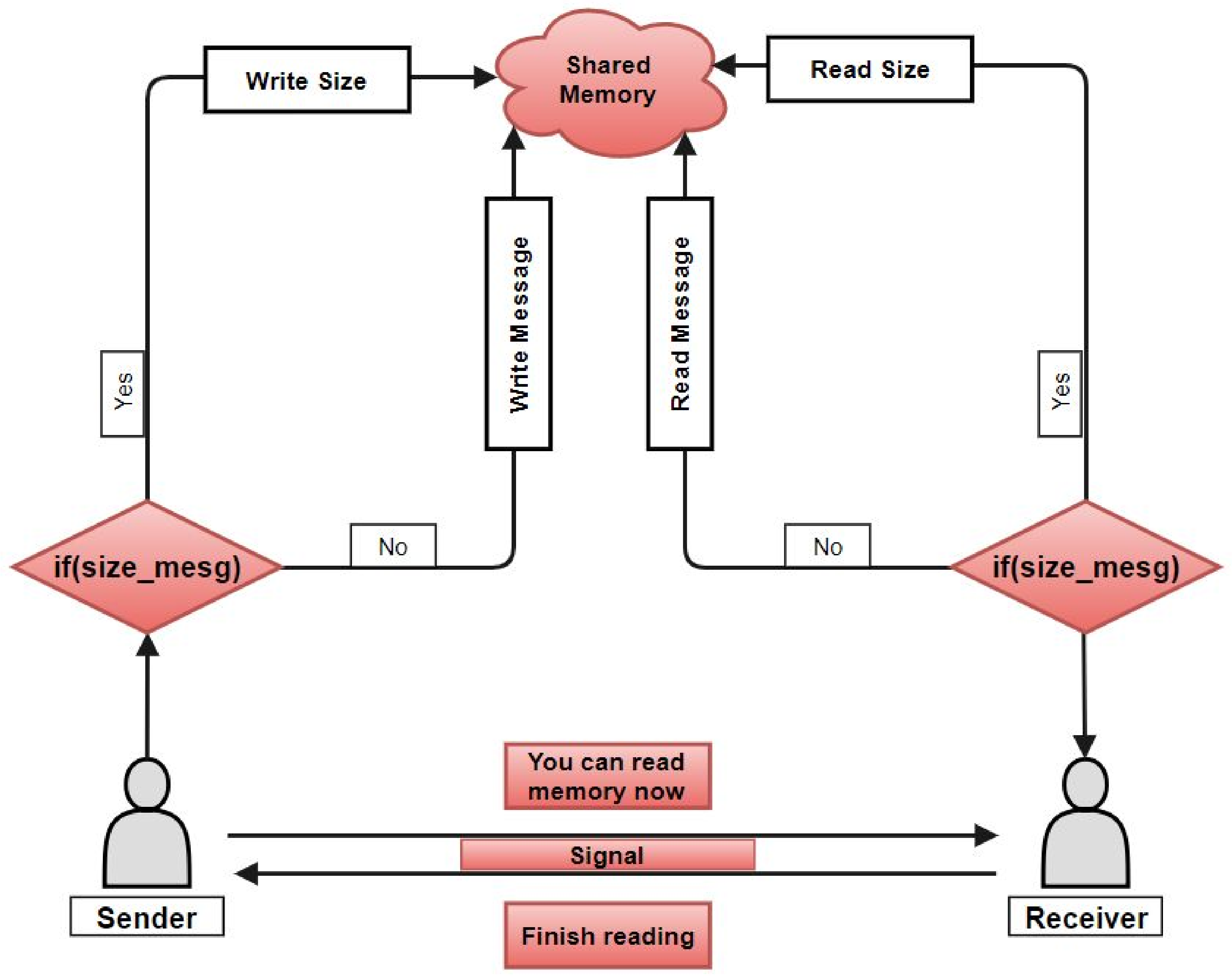
Shared Memory
Write Size
Read Size
Write Message
Read Message
Yes
Yes
No
No
if(size_mesg)
if(size_mesg)
You can read memory now
Signal
Finish reading
Sender
Receiver

Figure 1.3