Alexander Sigler

Dan Ortiz

Angad Singh

Jesus Guzman-Torres

CPSC 351-03

April 12, 2019

Assignment #1 Design

For Assignment 1, we had to implement an application that would synchronously transfer files between two processes. To create this application, we implemented two related programs; a sender program and a receiver program. The purpose of this assignment was to help familiarize ourselves with the concept of Interprocess communication. In order to implement two synchronous processes, we had to make use of shared memory, message queues and signals. The main idea of this assignment is to combine the principles of shared memory and message queues to create an application which sends a file and also processes the received file. To accomplish this task, we had to separately program a sender file and a receiver file to work concurrently.

The first part we needed for this project was the maker file. We used makefile to make comades that can be use in the command line to set up our code before we run the sender and the receiver.

To realize the actual application, we first started with the receiver program. First, we made some variables to represent the segment of shared memory and the message queue. We completed the initialization function which either checks for a file called

“keyfile.txt” or it creates a new file called “keyfile.txt” and fills it with some text. Next, we had to allocate a piece of shared memory and return the ID of that shared memory. Next, we attached our file to the shared memory using a pointer called sharedMemPtr. Next, we made sure to create a message queue and store the ID of that queue. All of these items compose the init function of the receiver program. The next part of the receiver program is the main loop where most of the actual interprocess communication occurs for this part of the application.

In the main loop of the program, we first opened the file so that it could be written into. Next, we had a do-while loop which would take care of of the actual receiving and saving of the shared memory. This do-while loop checks the sender to see if it has sent all of the data based on the size of the chunk of data. Once the data is received, then it is saved to the file. Still within the same loop, if the data is done receiving, then it pings the sender that it is ready for the next chunk of data. Finally, after the main loop, we utilized some functions; cleanUp, ctrlCSignal and main. In the main function, we just called all of the functions to make the program function with its counterpart.

After first running the receiver code we then got to work on the sender code. We started with initializing variables for the size of the shared memory size, ids for shared memory segment, message queue ,and a pointer to the shared memory. We split up the code into four functions including our main function. The main starts off checking the for the command line. Then starts initializing, by jumping to the init function, the rest of the program almost exactly like the receiver program checking if we have a keyfile.txt

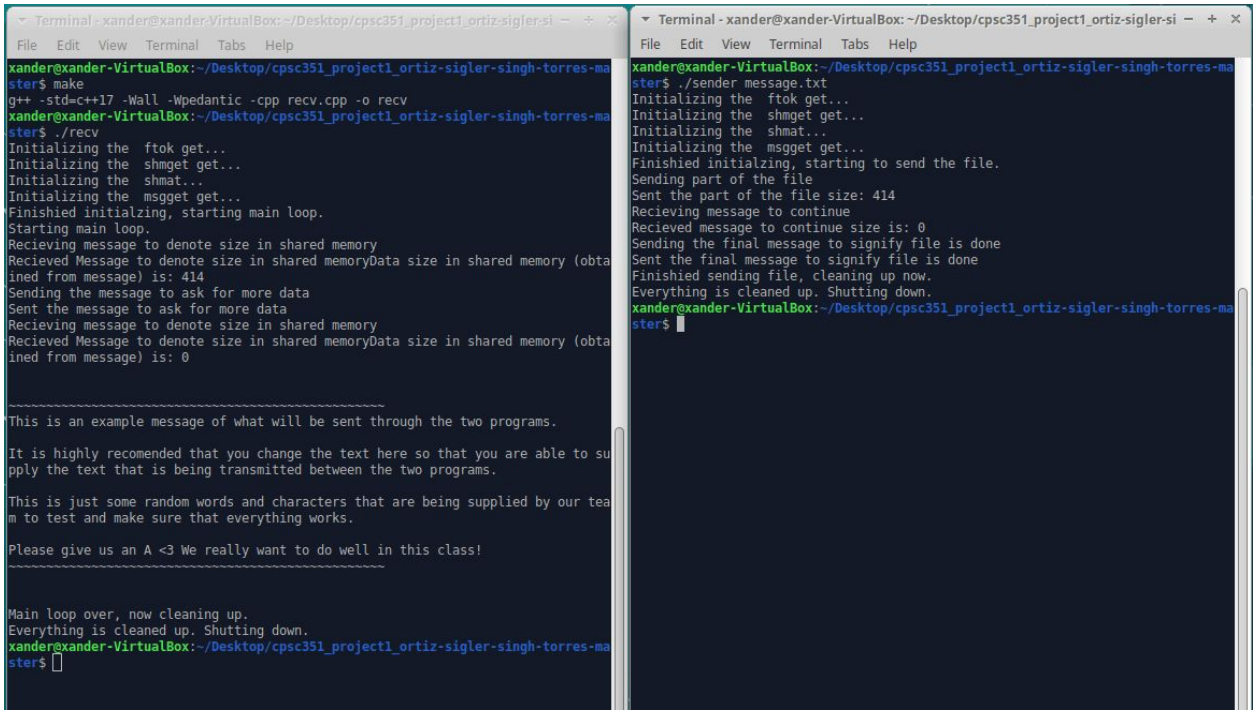
file if not it makes one, finding the id of shared memory , attaching file using pointer ,and create queue and add ids to the queue. By doing all this it connects to the shared memory and the message queue.

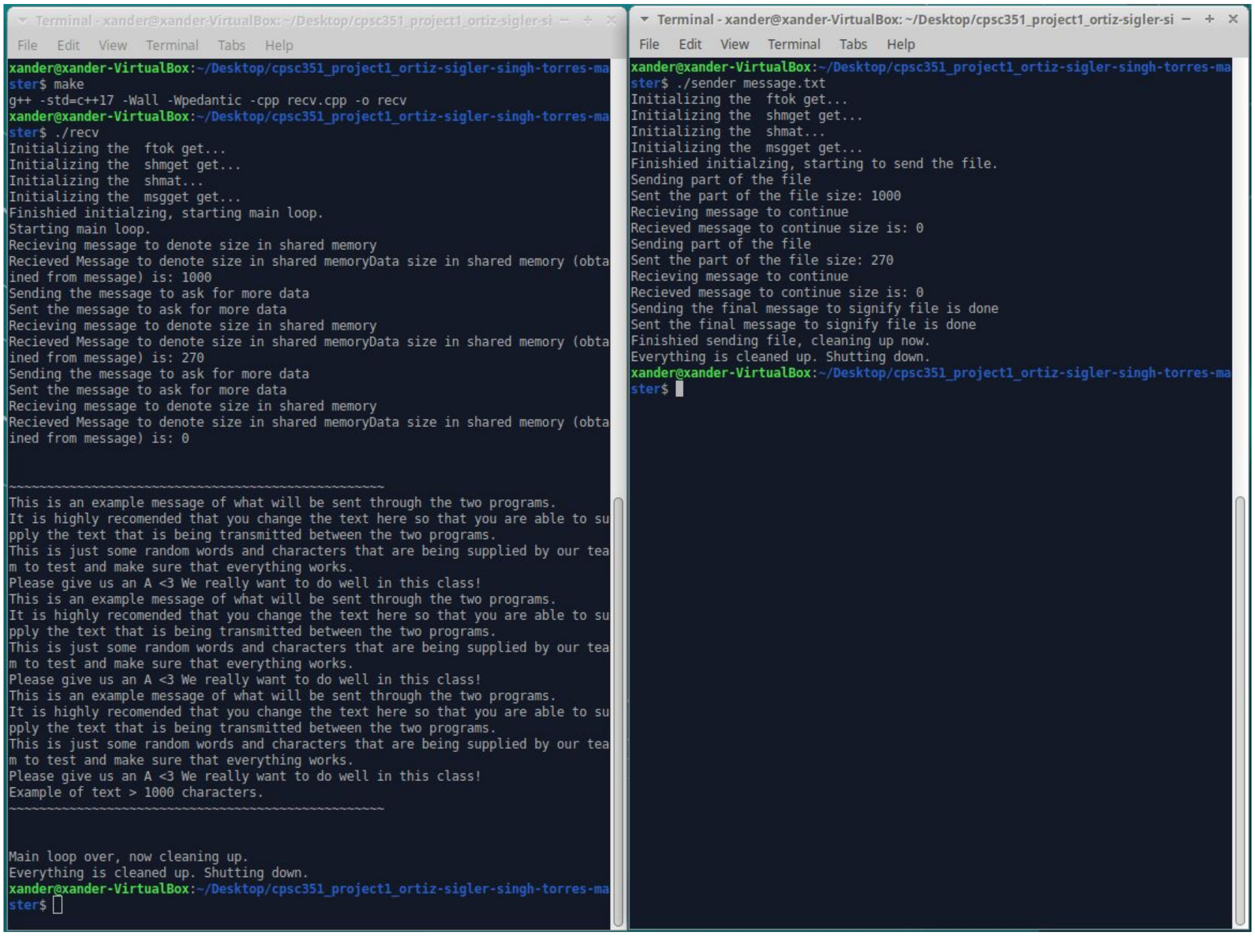
After all this is done is starts to send the file. Jumping to the send function it which takes in a filename. It opens the file for reading and checks if it was opened. It then it initializes variables that create a buffer to store the message. It then starts a loop to read the SHARED_MEMORY_CHUNK_SIZE of the file and save it into the shared memory. It then sends a message to the receiver saying the data is ready. Then it sends part of the flie and then sends part of the file size. It waits for receiver to send a message saying that it finished saving the memory chunk. When that is done it sends the file and ends the loop. After the loop it tells the the receiver it is done sending. Finally we run the cleanup function which detaches from the shared memory. We do all this by printing the steps to show the user what is going on in the code. We do all this by printing the steps to show the user what is going on in the code. We do all this by printing the steps to show the user what is going on in the code.

As we were completing this assignment, we came across a few issues that we difficult to overcome.Overall some of our group members had not had the best past experience working with the command line so teaching them how to use it was our first hurdle.While working on the actual code , one of the difficult parts of this application was opening the shared memory and the message queue. At first, we did not realize that there were specific flags like IPC_CREAT which needs to be in the receiver. This is important because the receiver initiates first so it has to create the section first. Another

issue that we encountered was sending the final message in the sender file. We knew that the size was supposed to be zero se we just sent a zero in the msgsend. In reality the value of zero should have been in the message type and not in the actual msgsend function. Because we sent the incorrect size, the receiver file never actually got the message to end the program. Once we sent a message with the size of zero, we were able to get it to work properly. Although we encountered some difficult to solve issues, we were able to work through them and gain a better understanding of inter-process communication.

In conclusion, this lab really helped us visualize and see what we are learning in class about memory sharing and message queues. We can see how interprocess communication can be applied in our different projects when it is needed. Besides the technical knowledge that we put into our code we learned how to work better as part of a group, something we will be doing a lot in our future.


Terminal - xander@xander-VirtualBox: ~/Desktop/cpsc351_project1_ortiz-sigler-si
File Edit View Terminal Tabs Help
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$ make
g++ -std=c++17 -Wall -Wpedantic -cpp recv.cpp -o recv
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$ ./recv
Initializing the ftok get...
Initializing the shmget get...
Initializing the shmat...
Initializing the msgget get...
Finishied initialzing, starting main loop.
Starting main loop.
Recieving message to denote size in shared memory
Recieved Message to denote size in shared memoryData size in shared memory (obtained from message) is: 414
Sending the message to ask for more data
Sent the message to ask for more data
Recieving message to denote size in shared memory
Recieved Message to denote size in shared memoryData size in shared memory (obtained from message) is: 0
This is an example message of what will be sent through the two programs.
It is highly recomended that you change the text here so that you are able to supply the text that is being transmitted between the two programs.
This is just some random words and characters that are being supplied by our team to test and make sure that everything works.
Please give us an A <3 We really want to do well in this class!
Main loop over, now cleaning up.
Everything is cleaned up. Shutting down.
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$
Terminal - xander@xander-VirtualBox: ~/Desktop/cpsc351_project1_ortiz-sigler-si
File Edit View Terminal Tabs Help
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$ ./sender message.txt
Initializing the ftok get...
Initializing the shmget get...
Initializing the shmat...
Initializing the msgget get...
Finishied initialzing, starting to send the file.
Sending part of the file
Sent the part of the file size: 414
Recieving message to continue
Recieved message to continue size is: 0
Sending the final message to signify file is done
Sent the final message to signify file is done
Finishied sending file, cleaning up now.
Everything is cleaned up. Shutting down.
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$


Terminal - xander@xander-VirtualBox: ~/Desktop/cpsc351_project1_ortiz-sigler-si
File Edit View Terminal Tabs Help
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$ make
g++ -std=c++17 -Wall -Wpedantic -cpp recv.cpp -o recv
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$ ./recv
Initializing the ftok get...
Initializing the shmget get...
Initializing the shmat...
Initializing the msgget get...
Finishied initialzing, starting main loop.
Starting main loop.
Recieving message to denote size in shared memory
Recieved Message to denote size in shared memoryData size in shared memory (obtained from message) is: 1000
Sending the message to ask for more data
Sent the message to ask for more data
Recieving message to denote size in shared memory
Recieved Message to denote size in shared memoryData size in shared memory (obtained from message) is: 270
Sending the message to ask for more data
Sent the message to ask for more data
Recieving message to denote size in shared memory
Recieved Message to denote size in shared memoryData size in shared memory (obtained from message) is: 0
This is an example message of what will be sent through the two programs.
It is highly recomended that you change the text here so that you are able to supply the text that is being transmitted between the two programs.
This is just some random words and characters that are being supplied by our team to test and make sure that everything works.
Please give us an A <3 We really want to do well in this class!
This is an example message of what will be sent through the two programs.
It is highly recomended that you change the text here so that you are able to supply the text that is being transmitted between the two programs.
This is just some random words and characters that are being supplied by our team to test and make sure that everything works.
Please give us an A <3 We really want to do well in this class!
This is an example message of what will be sent through the two programs.
It is highly recomended that you change the text here so that you are able to supply the text that is being transmitted between the two programs.
This is just some random words and characters that are being supplied by our team to test and make sure that everything works.
Please give us an A <3 We really want to do well in this class!
Example of text > 1000 characters.
Main loop over, now cleaning up.
Everything is cleaned up. Shutting down.
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$
Terminal - xander@xander-VirtualBox: ~/Desktop/cpsc351_project1_ortiz-sigler-si
File Edit View Terminal Tabs Help
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$ ./sender message.txt
Initializing the ftok get...
Initializing the shmget get...
Initializing the shmat...
Initializing the msgget get...
Finishied initialzing, starting to send the file.
Sending part of the file
Sent the part of the file size: 1000
Recieving message to continue
Recieved message to continue size is: 0
Sending part of the file
Sent the part of the file size: 270
Recieving message to continue
Recieved message to continue size is: 0
Sending the final message to signify file is done
Sent the final message to signify file is done
Finishied sending file, cleaning up now.
Everything is cleaned up. Shutting down.
xander@xander-VirtualBox:~/Desktop/cpsc351_project1_ortiz-sigler-singh-torres-master$