Operating Systems Project Report

Submitted

by

N. Jyothi Swarupa- AP19110010161

K. Sushma Rachel- AP19110010261

G. Rakshitha- AP19110010280

Ganta Jaiya- AP19110010246

M. Rishitha - AP19110010510

Department of Computer Science and Engineering

SRM University-AP, Andhra Pradesh, India

July - 2021

ACKNOWLEDGEMENT

In the accomplishment of completion of our project on Interprocess communication with client and server using shared memory, we would like to convey our special gratitude towards our professor Mr. Ashu Abdul, of Computer Science and Engineering. Your valuable guidance and suggestions helped us in various phases of the completion of this project. We will always be thankful to you in this regard.

We would like to appreciate our work that we have done in our project and thank each other for continuous support and coordination.

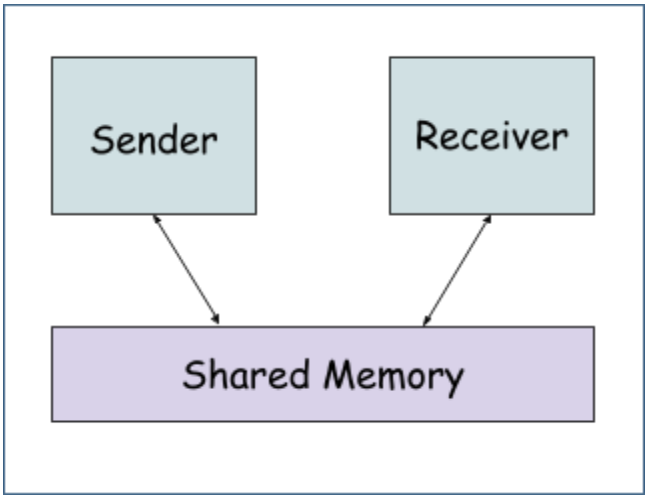
TABLE OF CONTENTS

I. INTRODUCTION

This project uses Inter Process Communication through shared memory. It is where two or more processes can access a common memory and the changes made by one process can be viewed by the other. The shared memory is created and the data is sent to the shared memory and from there data is sent to the sender. Here, the data in the given file is shared to the sender and prints the data.

II. OBJECTIVE OF THE PROJECT

The objective of the project is to implement Interprocess communication between two processes(i.e. sender and receiver) using shared memory. A file is sent from sender to receiver through a shared memory.



III. OVERVIEW OF THE PROJECT

At the sender side, a file is taken as input from the user and the data present in the file is sent to an array. A text variable is used to send the data from the array to the shared memory. A shared memory is created with IPC_CREAT | 0666 where 0666 sets the access permissions of the memory segment while IPC_CREAT tells the system to create a new memory segment for the shared memory. A pointer is used to attach with server address space. It then waits for the receiver to complete its work. Finally the pointer is detached and the shared memory is deleted.

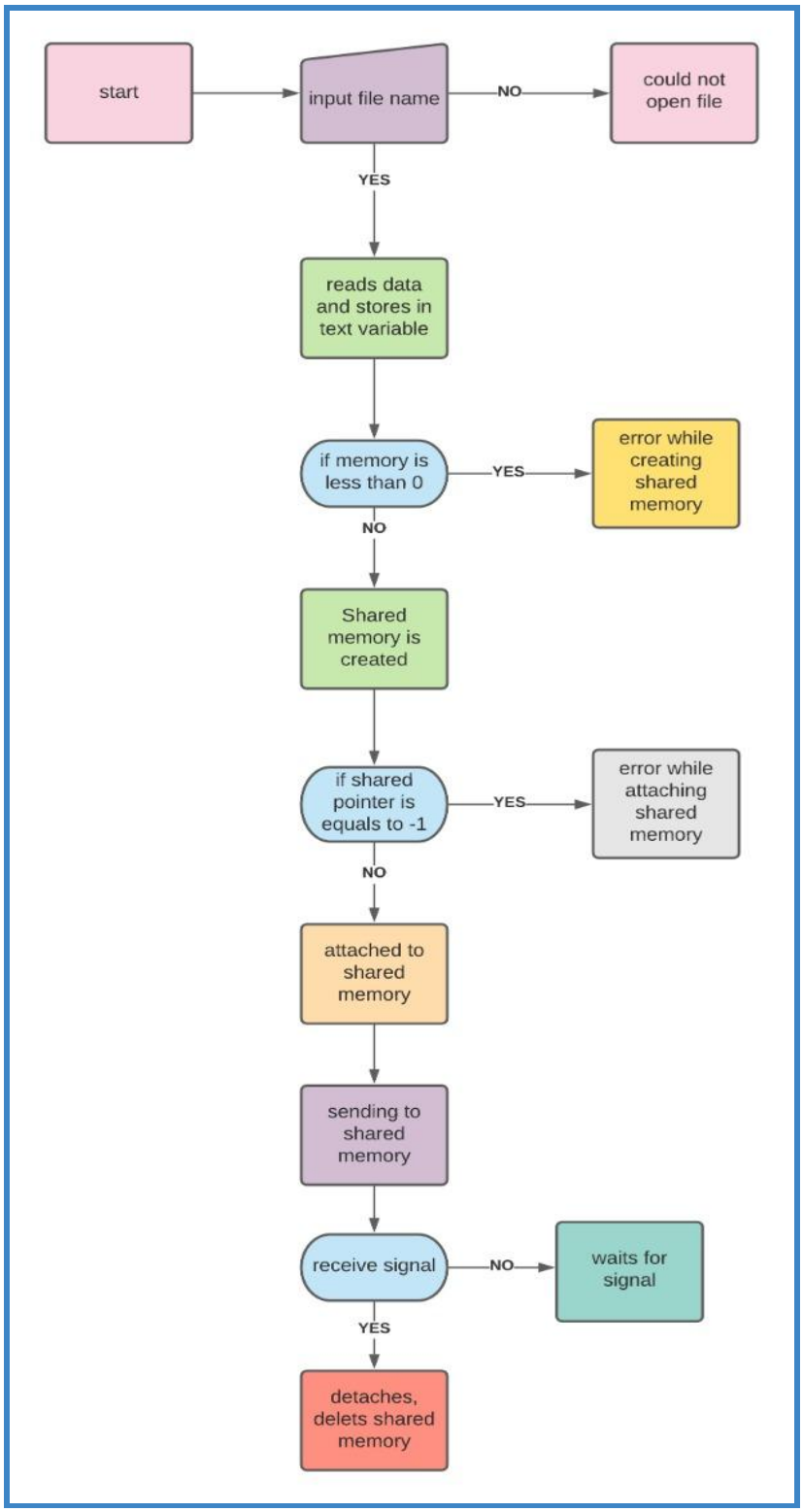
At the receiver side asks for the shared memory. A pointer is used to attach with the shared memory. The receiver uses the memory and signal server that the work is done. Finally it detaches the shared memory.

System calls used by sender and receiver:

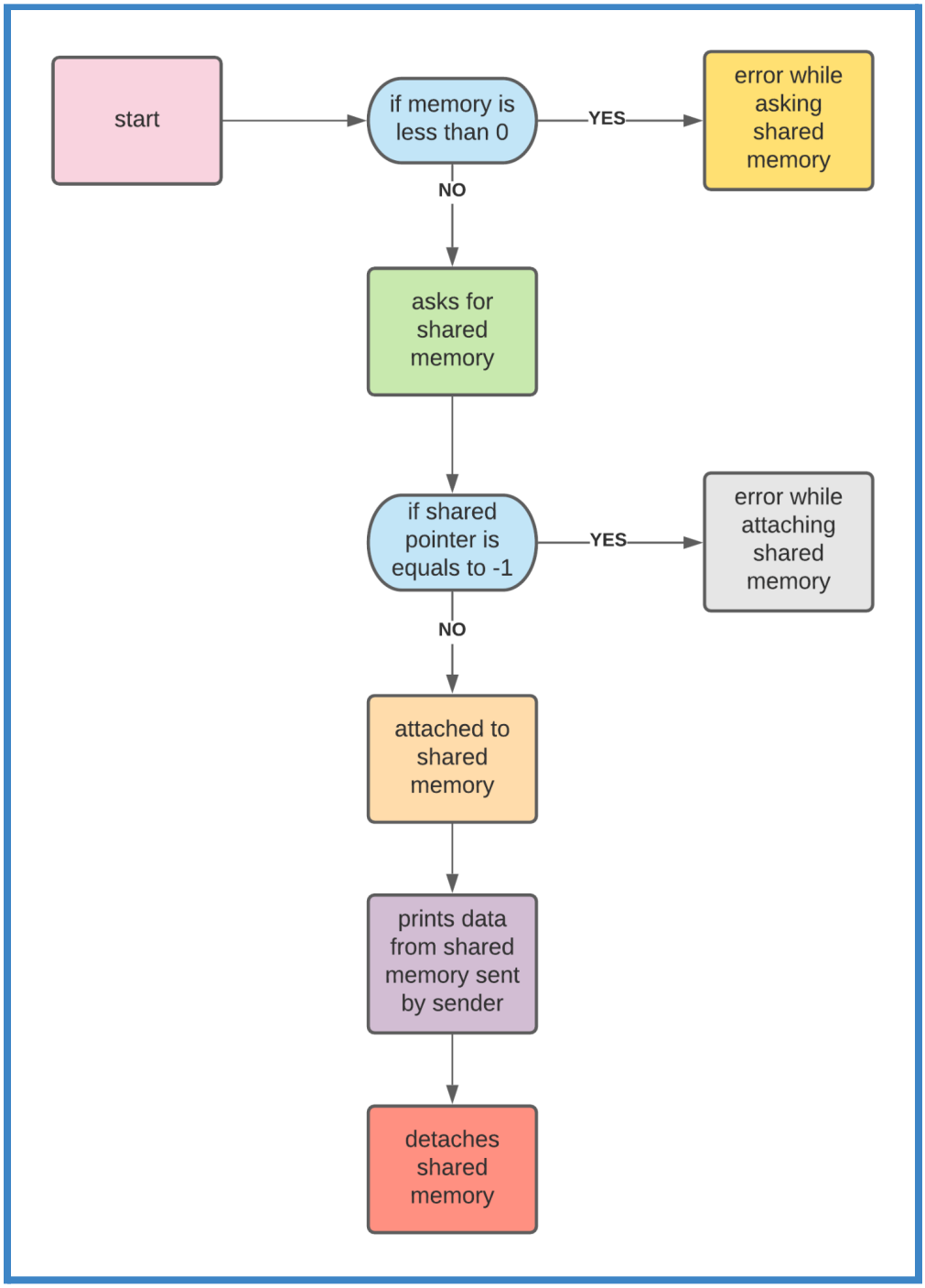
- shmget(): returns the shared identifier associated with key
- shmat(): Attaches the shared memory segment to a process, so that the memory contents will be accessed.
- shmdt(): used to detach a shared memory
- shmctl(): control the shared memory segment specified by the SharedMemoryID

IV. FLOW CHART

Sender:



Receiver:


start
if memory is less than 0
YES
error while asking shared memory
NO
asks for shared memory
if shared pointer is equals to -1
YES
error while attaching shared memory
NO
attached to shared memory
prints data from shared memory sent by sender
detaches shared memory

V. CODE

Sender:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/ipc.h>
#include<sys/types.h>
#include<string.h>
#include<sys/shm.h>

#define size 500

int main()
{
   int SharedMemoryID;
  char *sm_ptr, *t, *text;
  key_t key;
  key = 1234;

  char fname[100]="";
  printf("\nPlease Enter the filename: \n");
  scanf("%s",fname);
  FILE *fp = fopen(fname, "r");
```

```
if (fp == NULL)
{
  printf("Error: could not open file %s", fname);
      return 1;
     }

   const unsigned maxlength = 1000;
   char buffer[maxlength];

   while (fgets(buffer, maxlength, fp))
   {
      text=buffer;
   }
//Create Shared Memory
SharedMemoryID = shmget(key, size, IPC_CREAT | 0666);
if(SharedMemoryID < 0)
{
   printf("Error in creating shared memory\n");
   exit(0);
}

//Attach Shared Memory
sm_ptr = shmat(SharedMemoryID,NULL,0);
if(sm_ptr == (char *) -1)
```

```
{
    printf("Error in attaching shared memory\n");
}
//Sending text to shared Memory
t = sm_ptr;
int length = strlen(text);
for(int i=0;i<length;i++)
{
    *t = text[i];
     t++;
}

*t = 0;
while (*sm_ptr!='#')
    sleep(1);
//Detach Shared memory
shmdt(sm_ptr);

//Delete shared memory
shmctl(SharedMemoryID, IPC_RMID, NULL);
fclose(fp);
return 0;
}
```

Receiver:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/ipc.h>
#include<sys/types.h>
#include<string.h>
#include<sys/shm.h>

#define size 500

int main()
{
   int SharedMemoryID;
  char *sm_ptr, *t;
  key_t key;

  key = 1234;

  //Asking for shared memory
  SharedMemoryID=shmget(key, size, 0666);
  if(SharedMemoryID < 0)
  {
```

```
    printf("Error while asking shared memory\n");
    exit(0);
  }

  //Attach shared memory
  sm_ptr = shmat(SharedMemoryID, NULL, 0);
  if(sm_ptr==(char *) -1)
  {
    printf("Error attaching shared memory\n");
    exit(0);
  }

  t = sm_ptr;
  while(*t != 0)
  {
    printf("%c", *t);
    t++;
  }
  printf("\n");
  *sm_ptr = '#';
  //Detach shared memory
  shmdt(sm_ptr);
  return 0;
}
```

VI. OUTPUT

```
rakshitha@rakshitha-VirtualBox:~$ cd Desktop
rakshitha@rakshitha-VirtualBox:~/Desktop$ gcc -o a sender.c
rakshitha@rakshitha-VirtualBox:~/Desktop$ ./a

Please Enter the filename:
text.txt
rakshitha@rakshitha-VirtualBox:~/Desktop$
```

Printing the data that is sent by the sender which is in the shared memory

```
rakshitha@rakshitha-VirtualBox:~/Desktop$ gcc -o b receiver.c
rakshitha@rakshitha-VirtualBox:~/Desktop$ ./b
A paragraph is a series of related sentences developing a central idea, called the topic. Try to think about paragraphs in
 terms of thematic unity: a paragraph is a sentence or a group of sentences that supports one central, unified idea. Parag
raphs add one idea at a time to your broader argument.

rakshitha@rakshitha-VirtualBox:~/Desktop$
```

VII. CONCLUSION

Here we are sending the data in the file to shared memory and the receiver accesses the data . Advantage of shared memory is that the copying of message data is eliminated. It also speeds up the computational power of the system.