**Practical : 7**

* **Aim :**

Write the following programs using inter process communication – shared memory: [A] The program ‘writer.c’ will write 1 to 100 in the shared memory region. [B] Another program ‘reader.c’ that will read all the numbers from shared memory to make addition of it and display it.

* **Code :**

**Writer.c :**

#include<sys/shm.h>

#include<sys/ipc.h>

#include<stdio.h>

int main()

{

int sum=0,j=0;

key\_t key=ftok("shmfile",65);

int shmid=shmget(key,1024,0666|IPC\_CREAT);

int \*i=(int\*)shmat(shmid,0,0);

printf("\nWrite data in shared memory : \n");

for(j=1;j<=100;j++){

\*i=j;

printf("%d",\*i);

\*i++;

}

\*i=123;

shmdt(i);

printf("\n");

return 0;

}

**Reader.c :**

#include<sys/shm.h>

#include<sys/ipc.h>

#include<stdio.h>

int main()

{

int sum=0,j=0;

key\_t key=ftok("shmfile",65);

int shmid=shmget(key,1024,0666|IPC\_CREAT);

int \*i=(int\*)shmat(shmid,(void\*)0,0);

//printf("\nData read from memory : %d\n",\*i);

for(j=0;j<100;j++){

sum=sum+\*i++;

//printf("%d",i);

}

printf("\nSum : %d",sum);

shmdt(i);

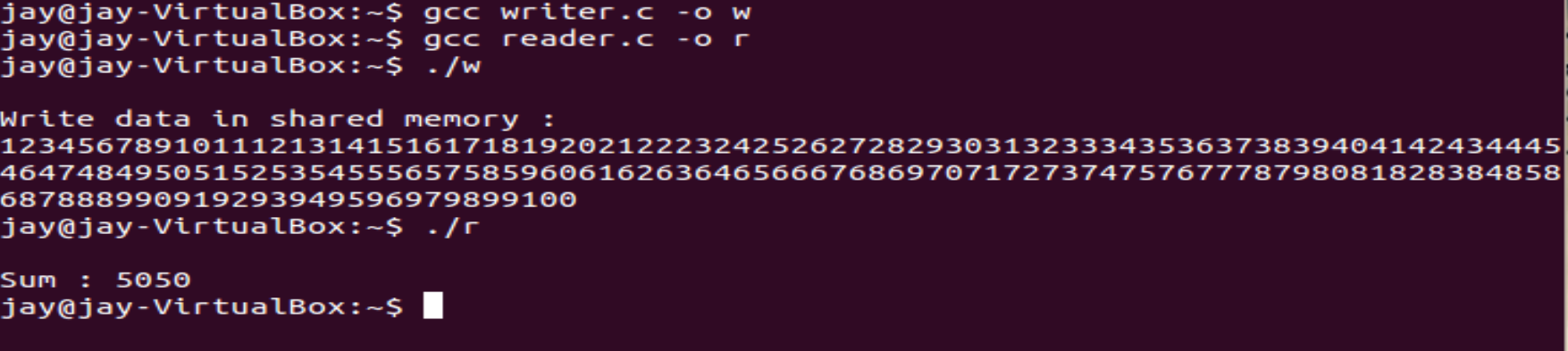
shmctl(shmid,IPC\_RMID,NULL);

printf("\n");

return 0;

}

* **Output :**



**Task :**

* **Aim :**

Create a shared memory that is going to be shared between 1 parent process and 3 child process. Parent process writes a random number in shared memory. After that following operations shold be done in strict following order. 1) child 1 reads the random number and mutliply with 10 and writes again in the shared memory (the number written by parent process should be removed) . 2) child 2 reads the number written by child 1 and perform modulo by 5 operation and writes again in the shared memory (the number written by child 1 should be removed). 3) Child 3 reads the number from shared memory and perform addition by last three digit of your enrollment number and writes again in the shared memory (the number written by child 2 should be removed). Print the final result on terminal.

* **Code :**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/shm.h>

#include<sys/ipc.h>

#include<stdio.h>

#include<stdlib.h>

int main(){

int p1,p2,p3;

int sum=0,j=0;

int num=rand()%2\*10;

key\_t key=ftok("shmfile",65);

int shmid=shmget(key,1024,0666|IPC\_CREAT);

int \*i=(int\*)shmat(shmid,0,0);

\*i=num;

printf("\nWrite data in shared memory : %d\n",num);

shmdt(i);

p1=fork();

if(p1==0){

sleep(1);

printf("\nChild 1 : ");

key\_t key=ftok("shmfile",65);

int shmid=shmget(key,1024,0666|IPC\_CREAT);

int \*num=(int\*)shmat(shmid,(void\*)0,0);

printf("\n\tData read from memory : %d",\*num);

\*num=(\*num)\*10;

printf("\n\tData write in memory : %d",\*num);

shmdt(i);

}

else{

p2=fork();

if(p2==0){

sleep(2);

printf("\nChild 2 :");

key\_t key=ftok("shmfile",65);

int shmid=shmget(key,1024,0666|IPC\_CREAT);

int \*num=(int\*)shmat(shmid,(void\*)0,0);

printf("\n\tData read from memory : %d",\*num);

\*num=(\*num)%5;

printf("\n\tData write in memory : %d",\*num);

shmdt(i);

}

else{

p3=fork();

if(p3==0){

sleep(3);

printf("\nChild 3 :");

key\_t key=ftok("shmfile",65);

int shmid=shmget(key,1024,0666|IPC\_CREAT);

int \*num=(int\*)shmat(shmid,(void\*)0,0);

printf("\n\tData read from memory : %d",\*num);

\*num=(\*num)+77;

printf("\n\tData write in memory : %d",\*num);

printf("\n");

shmdt(i);

shmctl(shmid,IPC\_RMID,NULL);

}

else{

sleep(3);

//printf("\nparent : %d\n",getpid());

}

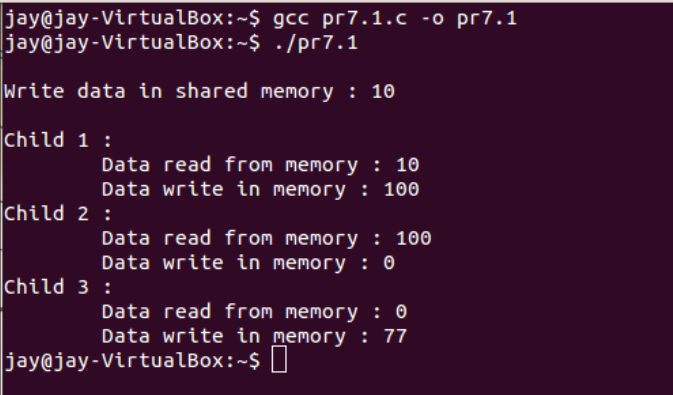
}

}

return 0;

}

* **Output :**

****

* **Conclusion :**
* In this practical we have learned about ipc using shared memory and also implemented task that are given.

**Practical : 8**

* **Aim :**

Create a message passing queue. Suppose a Process 'A' sends five message in following strict order. 1) Unix 2) Linux 3)MAC 4)Windows 5)Android. The Process 'B' should retrieve messages in following order. 1) MAC 2) Android 3)Linux 4) Windows 5)Unix

* **Code :**

**Process A :**

#include <iostream>

#include <sys/ipc.h>

#include <sys/msg.h>

using namespace std;

struct msg\_buffer {

long msg\_type;

char msg[100];

} message[5];

main() {

key\_t my\_key;

int msg\_id;

my\_key = ftok("progfile", 65);

msg\_id = msgget(my\_key, 0666 | IPC\_CREAT);

for(int i=0;i<5;i++)

{

message[i].msg\_type = 1;

cout<<"enter message "<<i<<" : ";

cin>>message[i].msg;

msgsnd(msg\_id, &message[i], sizeof(message[i]), 0);

}

}

**Process B :**

#include <iostream>

#include <sys/ipc.h>

#include <sys/msg.h>

using namespace std;

struct msg\_buffer {

long msg\_type;

char msg[100];

} message[5];

main() {

key\_t my\_key;

int msg\_id;

my\_key = ftok("progfile", 65);

msg\_id = msgget(my\_key, 0666 | IPC\_CREAT);

int o[5]={2,4,1,3,0};

string m[5];

for(int i=0;i<5;i++){

msgrcv(msg\_id, &message[i], sizeof(message[i]), 1, 0);

}

for(int i=0;i<5;i++){

cout<<"\t"<<message[o[i]].msg<<"\n";}

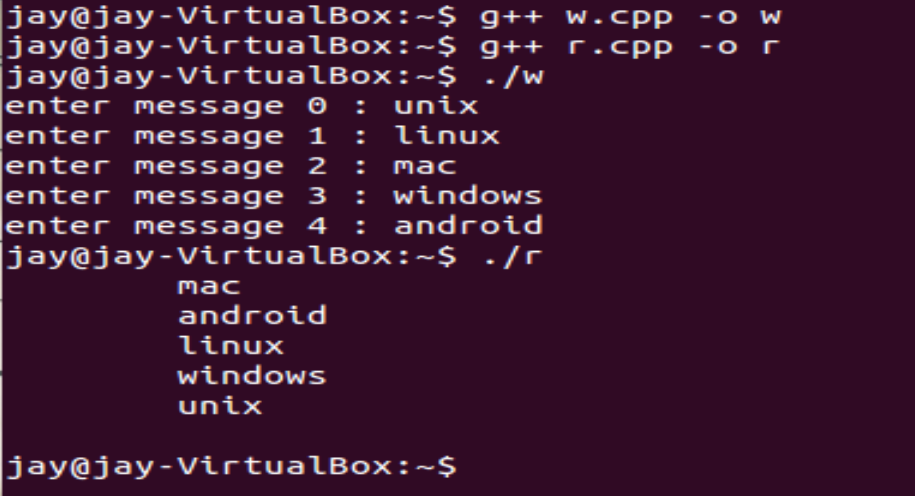
cout<<endl;

msgctl(msg\_id,IPC\_RMID,NULL);

return 0;

}

* **Output :**



* **Conclusion :**

In this practical we have learned about IPC using message passing and also implemented it.