VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
Permanently Affiliated to JNTU Kakinada, Approved by AICTE
Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified
Nambur, Pedakakani (M), Guntur (Dt) - 522508
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
B.Tech Program is Accredited by NBA

CERTIFICATE

Name of the Lab : LINUX PROGRAMMING

Name of the Student : Thota Nagababu

Student Regd. No. : 18BQ1A05K3

CLASS : III B.TECH. I SEM CSE – D

GIT HUB LINK: https://github.com/nagababuthota984/5K3-OS-LAB

INDEX

EXPERIMENT NO: 5 (a)

AIM : Write a C program illustrating two processes communicating using shared memory

DESCRIPTION :

Inter Process Communication through shared memory is a concept where two or more process can access the common memory. And communication is done via this shared memory where changes made by one process can be viewed by another process.

In this IPC model, a shared memory region is established which is used by the processes for data communication. This memory region is present in the address space of the process which creates the shared memory segment.The processes who want to communicate with this process should attach this memory segment into their address space.

SYNTAX:

i) struct Memory {};

This creates a structure which holds an integer (status) and an integer array of size 4. **struct** is a keyword in C. Memory is the name of structure.

ii) void main(int argc, char* argv[]){}

This is the one and only function written in this program. All the functionalities will be done in this function.

iii)ftok(): is use to generate a unique key.

iv)shmget(): int shmget(key_t,size_tsize,intshmflg); upon successful completion, shmget() returns an identifier for the shared memory segment.

v)shmat(): Before you can use a shared memory segment, you have to attach yourself to it using shmat(). void *shmat(int shmid ,void *shmaddr ,int shmflg);

shmid is shared memory id. shmaddr specifies specific address to use but we should set it to zero and OS will automatically choose the address.

vi)shmdt(): When you’re done with the shared memory segment, your program should detach itself from it using shmdt(). int shmdt(void *shmaddr);

vii)shmctl(): when you detach from shared memory,it is not destroyed. So, to destroy shmctl() is used. shmctl(int shmid,IPC_RMID,NULL);

CODE:

Code for server:

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

#define NOT_READY -1
#define FILLED 0
#define TAKEN 1

struct Memory
{
    int status;
    int data[4];
};
void main(int argc, char* argv[])
{
    key_t ShmKEY;
    int ShmID;
    struct Memory *ShmPTR;
    //prepare for shared memory
    ShmKEY = ftok("/",'x');
    ShmID = shmget(ShmKEY,sizeof(struct Memory),IPC_CREAT|0666);
    ShmPTR = (struct Memory *)shmat(ShmID,NULL,0);
    ShmPTR->status = NOT_READY;
    for(int i=0;i<4;i++)
    ShmPTR->data[i]=atoi(argv[i+1]);
    ShmPTR->status=FILLED;
    while(ShmPTR->status!=TAKEN);
    sleep(1);
    shmdt((void *) ShmPTR);
    shmctl(ShmID,IPC_RMID,NULL);
    exit(0);

}
```

Code for client program.

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

#define NOT_READY -1
#define FILLED 0
#define TAKEN 1

struct Memory
{
    int status;
    int data[4];
};

void main(int argc, char* argv[])

{

    key_t ShmKEY;

    int ShmID;

    struct Memory *ShmPTR;

    //prepare for shared memory

    ShmKEY = ftok("/",'x');

    ShmID = shmget(ShmKEY,sizeof(struct Memory),IPC_CREAT|0666);

    ShmPTR = (struct Memory *)shmat(ShmID,NULL,0);


    while(ShmPTR->status!=FILLED);

    printf("%d %d %d %d\n",ShmPTR->data[0],ShmPTR->data[1],ShmPTR-
>data[2],ShmPTR->data[3]);

    ShmPTR->status=TAKEN;

    shmdt((void *) ShmPTR);

    exit(0);


}
```

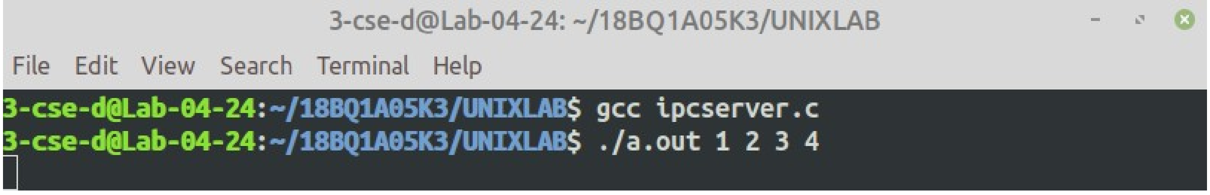
OUTPUT:

Server program will read the input given and stores it in the structure Memory.

Client program will access the structure Memory using shared memory concept and prints the data in the structure.

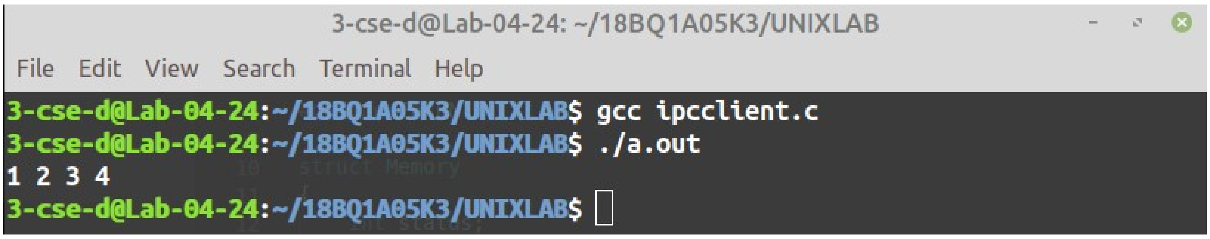
OUTPUT SCREEN SHOTS:

Screenshot for server program.


3-cse-d@Lab-04-24: ~/18BQ1A05K3/UNIXLAB
File Edit View Search Terminal Help

```
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$ gcc ipcserver.c
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$ ./a.out 1 2 3 4
```

Screenshot for client program.


3-cse-d@Lab-04-24: ~/18BQ1A05K3/UNIXLAB
File Edit View Search Terminal Help

```
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$ gcc ipcclient.c
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$ ./a.out
1 2 3 4
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$
```

EXPERIMENT NO: 5 (b)

AIM : Write a C program illustrating two processes communicating using shared memory

DESCRIPTION :

Inter Process Communication through shared memory is a concept where two or more process can access the common memory. And communication is done via this shared memory where changes made by one process can be viewed by another process.

In this IPC model, a shared memory region is established which is used by the processes for data communication. This memory region is present in the address space of the process which creates the shared memory segment.The processes who want to communicate with this process should attach this memory segment into their address space.

SYNTAX:

i)ftok(): is use to generate a unique key.

ii)shmget(): int shmget(key_t,size_tsize,intshmflg); upon successful completion, shmget() returns an identifier for the shared memory segment.

iii)shmat(): Before you can use a shared memory segment, you have to attach yourself to it using shmat(). void *shmat(int shmid ,void *shmaddr ,int shmflg);

shmid is shared memory id. shmaddr specifies specific address to use but we should set it to zero and OS will automatically choose the address.

iv)shmdt(): When you're done with the shared memory segment, your program should detach itself from it using shmdt(). int shmdt(void *shmaddr);

v)shmctl(): when you detach from shared memory,it is not destroyed. So, to destroy shmctl() is used. shmctl(int shmid,IPC_RMID,NULL);

CODE:

Code for reader:

```
#include <stdio.h>

#include <sys/ipc.h>

#include <sys/shm.h>




void main(void )

{

    key_t ShmKEY;

    int ShmID;


    ShmKEY = ftok("shmfile",'x');

    ShmID = shmget(ShmKEY,1024,IPC_CREAT|0666);


    char *str = (char *)shmat(ShmID,(void *)0,0);


    printf("Data read from memory is: %s\n",str);


    shmdt(str);

    shmctl(ShmID,IPC_RMID,NULL);


}
```

Code for writer:

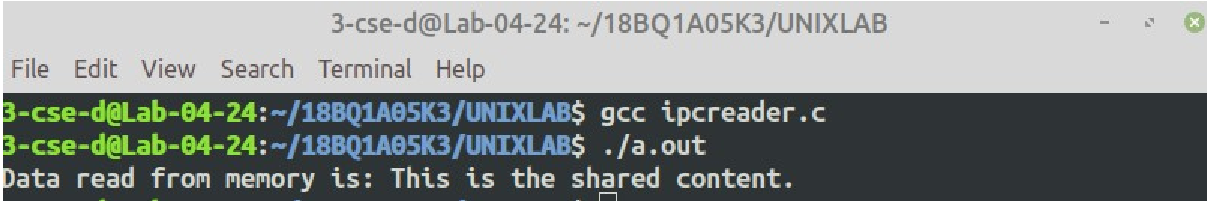
```
#include <stdio.h>

#include <sys/ipc.h>

#include <sys/shm.h>




void main(void )

{

    key_t ShmKEY;

    int ShmID;


    ShmKEY = ftok("shmfile",'x');

    ShmID = shmget(ShmKEY,1024,IPC_CREAT|0666);


    char *str = (char *)shmat(ShmID,(void *)0,0);


    printf("Write data: ");

    gets(str);

    printf("Data written in memory is: %s\n ",str);

    shmdt(str);


}
```

OUTPUT:

Reader will read a string which is to be shared. Writer will access that shared string and prints it. This is entirely based upon shared memory concept.

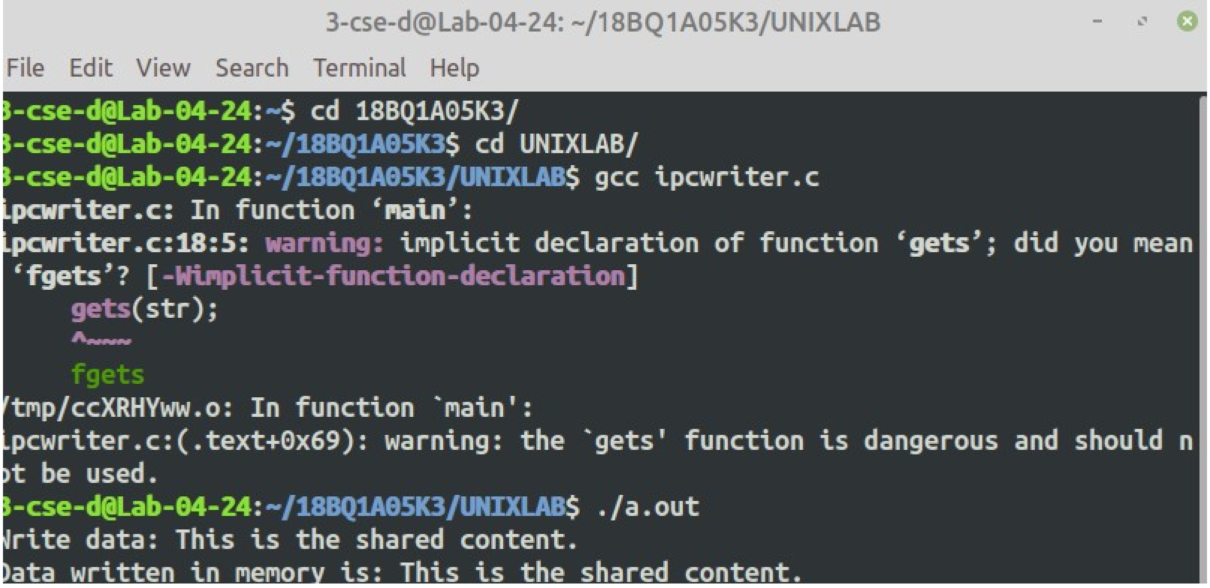
OUTPUT SCREENSHOTS:

SCREENSHOT FOR READER.


3-cse-d@Lab-04-24: ~/18BQ1A05K3/UNIXLAB
File Edit View Search Terminal Help

```
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$ gcc ipcreader.c
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$ ./a.out
Data read from memory is: This is the shared content.
```

SCREENSHOT FOR WRITER.


3-cse-d@Lab-04-24: ~/18BQ1A05K3/UNIXLAB
File Edit View Search Terminal Help

```
3-cse-d@Lab-04-24:~$ cd 18BQ1A05K3/
3-cse-d@Lab-04-24:~/18BQ1A05K3$ cd UNIXLAB/
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$ gcc ipcwriter.c
ipcwriter.c: In function ‘main’:
ipcwriter.c:18:5: warning: implicit declaration of function ‘gets’; did you mean
 ‘fgets’? [-Wimplicit-function-declaration]
     gets(str);
     ^~~~
     fgets
/tmp/ccXRHYww.o: In function `main':
ipcwriter.c:(.text+0x69): warning: the `gets' function is dangerous and should n
ot be used.
3-cse-d@Lab-04-24:~/18BQ1A05K3/UNIXLAB$ ./a.out
Write data: This is the shared content.
Data written in memory is: This is the shared content.
```