Exercise 3:

Implementation of Interprocess communication using shared memory.

Aim:

To Implement Interprocess communication using shared memory.

Description:

1.shm_open():

Header:

```
#include <sys/mman.h>
#include <sys/stat.h>       /* For mode constants */
#include <fcntl.h>          /* For O_* constants */
```

Syntax:

```
int shm_open(const char *name, int oflag, mode_t mode);
```

Description:

shm_open() creates and opens a new, or opens an existing, POSIX shared memory object. A POSIX shared memory object is in effect a handle which can be used by unrelated processes to mmap(2) the same region of shared memory. The shm_unlink() function performs the converse operation, removing an object previously created by shm_open().

2.shmget():

Header:

```
#include <sys/shm.h>
```

Syntax:

```
int shmget(key_t key, size_t size, int shmflg);
```

Description:

shmget() returns the identifier of the System V shared memory segment associated with the value of the argument key. It may be used either to obtain the identifier of a previously created shared memory segment (when shmflg is zero and key does not have the value IPC_PRIVATE), or to create a new set.

Exercise 3:

Implementation of Interprocess communication using shared memory.

3.shmat():

Header:

#include <sys/shm.h>

Syntax:

void *shmat(int shmid, const void *shmaddr, int shmflg);

Description:

The shmat() function attaches the shared memory segment associated with the shared memory identifier specified by shmid to the address space of the calling process. The segment is attached at the address specified by one of the following criteria:

Code:

InterProcSend.c

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <stdlib.h>
//typedef struct values{
//        char* something;
//        int val;
//}val;


int main(){
char* SharedMemName="Sender";
char* message="Hi There!!\n";
```

```
val* v1;
v1=(val*)malloc(sizeof(val));
v1->something=message;
v1->val=2334;
int shmFD;
void* ptr;
shmFD=shm_open(SharedMemName,O_CREAT|O_RDWR,0666);
if(shmFD==-1){
        write(1,"[+][shm_open]Failed To Create Shared Memory",40);
        return -1;
}
ftruncate(shmFD,4096);
ptr=mmap(0,4096,PROT_WRITE,MAP_SHARED,shmFD,0);
sprintf(ptr,"%s",message);
ptr+=strlen(message);
}
```

InterProcRecv.c:

```
#include <stdio.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <unistd.h>
int main(){
        char buf[10];
        char* SendShm="Sender";
        int shmFD;
        shmFD=shm_open(SendShm,O_RDONLY,0666);
        void* ptr=mmap(0,4096,PROT_READ,MAP_SHARED,shmFD,0);
        printf("%s",(char*)ptr);
```

```
        //read(shmFD,buf,10);
        //write(1,buf,10);
        shm_unlink("Sender");

}
```

OUTPUT:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103# gcc InterProcSend.c -lrt
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103# ./a.out
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103# gcc InterProcRecv.c -lrt
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103# ./a.out
Hi There!!
```

2.***IPCS1.c:***

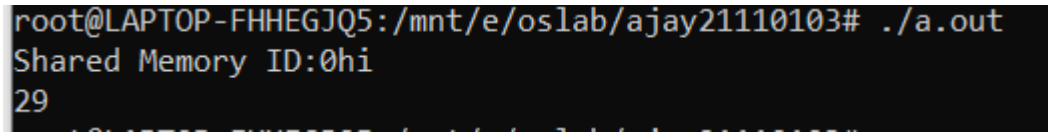
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/shm.h>
typedef struct test_shm{
        char* something;
        int value;

}t_s;

int main(){
key_t key=1234;
int id=shmget(key,1024,IPC_CREAT|0644);
printf("Shared Memory ID:%d",id);
t_s*ptr=shmat(id,NULL,0);
t_s *entry;
entry=(t_s*)malloc(sizeof(t_s));
```

Exercise 3:

Implementation of Interprocess communication using shared memory.

```
entry->something="hi";

entry->value=29;

memcpy(ptr,entry,sizeof(t_s));


printf("%s\n%d\n",ptr->something,ptr->value);


}
```

Output:



Result:

Thus Inter Process Communication was Established using shm.h library functions (shm_get,shmat).