Ex. No.: 7

Date: 28/3/25

IPC USING SHARED MEMORY

Aim:

To write a C program to do Inter Process Communication (IPC) using shared memory between sender process and receiver process.

Algorithm:

sender

- 1. Set the size of the shared memory segment
- 2. Allocate the shared memory segment using shunget
- 3. Attach the shared memory segment using shmat
- 4. Write a string to the shared memory segment using sprintf
- 5. Set delay using sleep
- 6. Detach shared memory segment using shmdt

receiver

- 1. Set the size of the shared memory segment
- 2. Allocate the shared memory segment using shmget
- 3. Attach the shared memory segment using shmat
- 4. Print the shared memory contents sent by the sender process.
- 5. Detach shared memory segment using shmdt

Program Code:

sender.c

include Lstolio. h>

include Lstallib. hs

include Lsys/spc.hs
include Lsys/shm.hs
include Lunistal.hs
include Lstring.hs

```
# define SIZE 1024
int main()?
    key_t key = 1234;
     int should;
     Char + shared - mem;
     Showid = Shonget (key, SIZE, Obbb ) IPC_CREAT);
     if (should == -1) &
             persor ("shinget failed");
              exit(i);
       Shared_mem = (char*) shimat (shimid, NULL, 0);
      4
      if (shared -mem == (chart) -1) &
               person ("shorat failed");
                exit(i);
       Sprintf (shared-menn, "Hello from Sender Process!");
        prints (" Sender wrote: /sln", shared _mem);
        3/20p(5);
         Shmat (shared -mem);
         return 0;
   J
```

```
receiver.c
#include Lstdio.hs
# include establib. hs
# include Lsyslipe.hs
# include Lsys/shm.h>
# include Lunistd. hs
# Include Lstring. h >
# define SIZE 1024
 int main()?
      Key-t key: 1284;
       int should;
       char * shared - mem;
       should = shonget (key, SIZE, Obbb);
            (showid : = -1) {
               pervor (" shinger failed");
               exit(1);
        3
         shoved-mem = (char*) Shorred (showed, NUL, 0);
             (shared - man = = (chax*) -1) {
                 pervior ("stimat failed");
                 exit(1);
         pourty ("Reciever read: 1.sln", should_mem);
         should ( should - mem);
          short (shortd, IPC_RMID, NULL);
           ochum o;
```

Sample Output

Terminal 1

[root@localhost student]# gcc sender.c -o sender [root@localhost student]# ./sender

Terminal 2

[root@localhost student]# gcc receiver.c -o receiver [root@localhost student]# ./receiver Message Received: Welcome to Shared Memory [root@localhost student]#

Result:

botween Sender process & recrever process was compered executed successfully.

Output:

Sender wrote: Hello from Sender Proces!

Raciever read: Hello from Sander Process!

Almostly pullbarte indest lound a transform at my is