Exp. No 7d **Shared Memory**

Date:

Aim

To demonstrate communication between process using shared memory.

Shared memory

- Two or more processes share a single chunk of memory to communicate randomly.
- Semaphores are generally used to avoid race condition amongst processes.
- Fastest amongst all IPCs as it does not require any system call.
- It avoids copying data unnecessarily.

Algorithm

Server

1. Initialize size of shared memory *shmsize* to 27.
2. Initialize *key* to 2013 (some random value).
3. Create a shared memory segment using shmget with *key* & IPC_CREAT as parameter.
   a. If shared memory identifier *shmid* is -1, then stop.
4. Display *shmid*.
5. Attach server process to the shared memory using shmmat with *shmid* as parameter.
   a. If pointer to the shared memory is not obtained, then stop.
6. Clear contents of the shared region using memset function.
7. Write a–z onto the shared memory.
8. Wait till client reads the shared memory contents
9. Detatch process from the shared memory using shmdt system call.
10. Remove shared memory from the system using shmctl with IPC_RMID argument
11. Stop

Client

1. Initialize size of shared memory *shmsize* to 27.
2. Initialize *key* to 2013 (same value as in server).
3. Obtain access to the same shared memory segment using same *key*.
   a. If obtained then display the *shmid* else print "Server not started"
4. Attach client process to the shared memory using shmmat with *shmid* as parameter.
   a. If pointer to the shared memory is not obtained, then stop.
5. Read contents of shared memory and print it.
6. After reading, modify the first character of shared memory to '*'
7. Stop

Program

Server

```
/* Shared memory server - shms.c */

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

#define shmsize 27

main()
{
   char c;
   int shmid;
   key_t key =  2013;
   char *shm, *s;

   if ((shmid = shmget(key, shmsize, IPC_CREAT|0666)) < 0)
   {
      perror("shmget");
      exit(1);
   }
   printf("Shared memory id : %d\n", shmid);

   if ((shm = shmat(shmid, NULL, 0)) == (char *) -1)
   {
      perror("shmat");
      exit(1);
   }

   memset(shm, 0, shmsize);
   s = shm;
   printf("Writing (a-z) onto shared memory\n");
   for (c = 'a'; c <= 'z'; c++)
      *s++ = c;
   *s = '\0';

   while (*shm != '*');
   printf("Client finished reading\n");

   if(shmdt(shm) != 0)
      fprintf(stderr, "Could not close memory segment.\n");

   shmctl(shmid, IPC_RMID, 0);
}
```

Client

```
/* Shared memory client - shmc.c */

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

#define shmsize 27

main()
{
   int shmid;
   key_t key = 2013;
   char *shm, *s;

   if ((shmid = shmget(key, shmsize, 0666)) < 0)
   {
      printf("Server not started\n");
      exit(1);
   }
   else
      printf("Accessing shared memory id : %d\n",shmid);

   if ((shm = shmat(shmid, NULL, 0)) == (char *) -1)
   {
      perror("shmat");
      exit(1);
   }

   printf("Shared memory contents:\n");
   for (s = shm; *s != '\0'; s++)
      putchar(*s);
   putchar('\n');

   *shm = '*';
}
```

Output

Server

```
$ gcc shms.c -o shms

$ ./shms
Shared memory id : 196611
Writing (a-z) onto shared memory
Client finished reading
```

Client

```
$ gcc shmc.c -o shmc

$ ./shmc
Accessing shared memory id : 196611
Shared memory contents:
abcdefghijklmnopqrstuvwxyz
```

Result

Thus contents written onto shared memory by the server process is read by the client process.