**Experiment III – Shared Memory & Semaphores**

Jyothiradithyan k

S3 CSE, 34.

**1. Implementation of Shared Memory**

There are **two** processes called **server** and **client** accessing the **same** shared memory. Write programs named ***server.c*** and ***client.c***. The ***server.c*** creates a shared memory, updates it with a string and waits for ***client.c*** for its updation. The ***client.c*** reads the same shared memory and updates with **reverse of the string**. Then, ***server.c*** exits from wait loop and print whether the string is **palindrome** or not.

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define SHMSIZE 27

void main(){

int shmid,shmid2,i;

key\_t key,key2;

char ch,\*shm,\*s,\*f;

char str[15];

key = 5678;

key2 = 5677;

if((shmid = shmget(key,SHMSIZE,IPC\_CREAT|0666))<0){ //create shared memory

perror("shmget");

exit(1);

}

if((shm = shmat(shmid,NULL,0))==(char\*)-1){ //attach segment

perror("shmat");

exit(1);

}

if((shmid2 = shmget(key2,SHMSIZE,IPC\_CREAT|0666))<0){ //create shared memory

perror("shmget");

exit(1);

}

if((f = shmat(shmid2,NULL,0))==(char\*)-1){ //attach segment

perror("shmat");

exit(1);

}

printf("Input string: "); //entering the word

s = shm;

scanf("%s",s);

printf("SERVER: Updated Shared Memory: \t%s\n",shm);

printf("Waiting for client..");

strcpy(str,s);

\*f = '0'; //waiting for client

while(\*f == '0');

printf("\nSERVER: Shared Memory: \t%s\n",shm);

i = strcmp(str,shm); //checking palindrome

if(i==0){

printf("It is a palindrome\n");

}

else

printf("It is not a palindrome\n");

}

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define SHMSIZE 27

char \*strrev(char \*str)

{

if (!str || ! \*str)

return str;

int i = strlen(str) - 1, j = 0;

char ch;

while (i > j)

{

ch = str[i];

str[i] = str[j];

str[j] = ch;

i--;

j++;

}

return str;

}

void main(){

int shmid,shmid2,i;

key\_t key,key2;

char ch,\*shm,\*s,\*f;

key = 5678;

key2 = 5677;

if((shmid = shmget(key,SHMSIZE,0666))<0){ //attach segment

perror("shmget");

exit(1);

}

if((shm = shmat(shmid,NULL,0))==(char\*)-1){ //create shared memory

perror("shmat");

exit(1);

}

if((shmid2 = shmget(key2,SHMSIZE,0666))<0){ //attach segment

perror("shmget");

exit(1);

}

if((f = shmat(shmid2,NULL,0))==(char\*)-1){ //create shared memory

perror("shmat");

exit(1);

}

printf("String from Shared Memory:\t");

s = shm;

printf("%s",s);

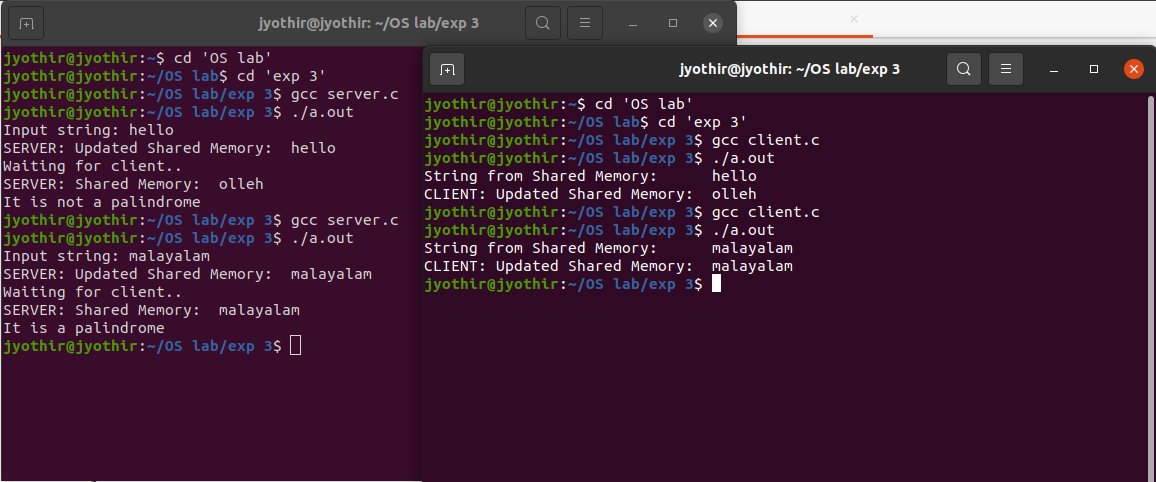
strrev(s); //string reversal

printf("\nCLIENT: Updated Shared Memory:\t");

printf("%s\n",s);

\*f = '1';

}



**2. Implementation of Semaphore**

Write a program named ***sem.c*** that creates **two** threads, ***thread1*** and ***thread2*** within the same process. The main program reads a string and creates ***thread1*** and ***thread2***. Thread1 in its critical section reverses the string and converts first letter to Uppercase, ***thread2*** in its critical section reads the string updated by ***thread1*** and reverses the string and converts first letter to Uppercase.

*Sample output of the program:*

***Main Process:*** Enter a string: **malayalam**

***Thread1:*** Entering *Critical Section*...

Reverse and first letter Uppercase: **Malayalam**

***Thread1:*** Exiting *Critical Section*...

***Thread2:*** Entering *Critical Section*...

Reverse and first letter Uppercase: **MalayalaM**

***Thread2:*** Exiting *Critical Section*...

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#include <string.h>

#include <ctype.h>

sem\_t mutex;

char str[15];

char \*strrev(char \*str)

{

if (!str || ! \*str)

return str;

int i = strlen(str) - 1, j = 0;

char ch;

while (i > j)

{

ch = str[i];

str[i] = str[j];

str[j] = ch;

i--;

j++;

}

return str;

}

void\* thread(void\* arg)

{

sem\_wait(&mutex); //wait

printf("Entering critical section...\n");

strrev(str); //critical section

str[0] = toupper(str[0]);

printf("Reverse and first letter uppercase: %s",str);

printf("\nExiting critical section...\n"); //signal

sem\_post(&mutex);

}

int main()

{

printf("Main process: Enter a string: ");

scanf("%s",str);

sem\_init(&mutex, 0, 1);

pthread\_t t1,t2;

printf("Thread1: ");

pthread\_create(&t1,NULL,thread,NULL);

sleep(2);

pthread\_create(&t2,NULL,thread,NULL);

pthread\_join(t1,NULL);

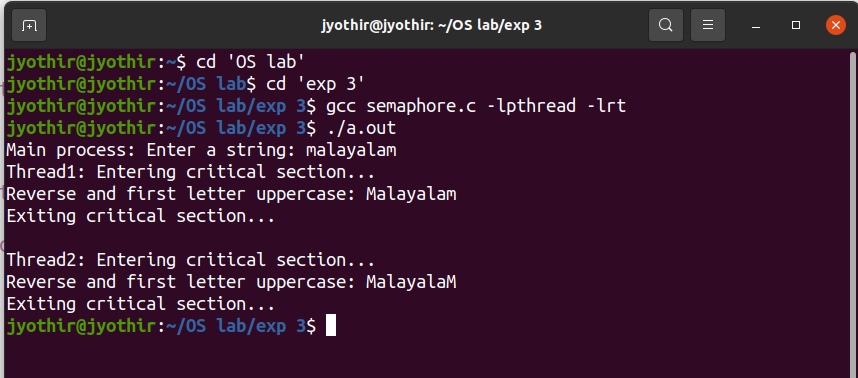
printf("\nThread2: ");

pthread\_join(t2,NULL);

sem\_destroy(&mutex);

return 0;

}

******