#include<pthread.h>

#include<stdio.h>

#include<semaphore.h>

#include<unistd.h>

void \*funl();

void \*fun2();

int shared=1; //shared variable

sem\_t s; //semaphore variable

int main()

{

sem\_init(&s,0,1); //initialize semaphore variable - Ist argument is address of variable, 2nd is number of

processes sharing semaphore, 3rd argument is the initial value of semaphore variable

pthread\_t thread1, thread2;

pthread\_create(&thread1, NULL, funl, NULL);

pthread\_create(&thread2, NULL, fun2, NULL);

pthread\_join(thread1, NULL);

pthread\_join(thread2,NULL);

printf("Final value of shared is %d\n",shared); //prints the last updated value of shared variable

}

void \*fun!()

{

int x;

sem\_wait(&s); //executes wait Operation on s

x = shared;//thread] reads value of shared variable

printf("Thread! reads the value as %d\n",x);

x++; //thread! increments its value

print("Local updation by Thread1; %d\n",x);

sleep(1); (bread | is preempted by thread 2

shared=x; //thread one updates the value of shared variable

printf("Value of shared variable updated by Thread 1 is: %d\n",shared);

sem\_post(&s);

}

void \*fun2()

{

int y;

sem\_wait(&s);

y=shared;//thread2 reads value of shared variable

printf("Thread2 reads the value as %d\n",y);

y--; /thread2 increments its value

printf("Local updation by Thread2: %d\n",y);

sleep(1); /Ahread2 is preempted by thread |

shared=y; /Ahread2 updates the value of shared variable

printf("Value of shared variable updated by Thread2 is; %d\n",shared);

sem\_post(&s);

}

**Output:**

Thread1 reads the value as 1

Local updation by Thread1: 2

Value of shared variable updated by Thread1 is: 2

Thread2 reads the values as 2

Local updation by Thread2: 1

Value of shared variable updated by Thread2 is: 1

Final values of shared is 1