Lab-04

**Write a C program to simulate producer-consumer problem using semaphores.**

#include<stdio.h>

#include<stdlib.h>

int mutex=1,full=0,empty=3,x=0;

int main()

{

    int n;

    void producer();

    void consumer();

    int wait(int);

    int signal(int);

    printf("\n1.Producer\n2.Consumer\n3.Exit");

    while(1)

    {

        printf("\nEnter your choice: ");

        scanf("%d",&n);

        switch(n)

        {

            case 1: if((mutex==1)&&(empty!=0))

                    producer();

                    else

                    printf("Buffer is full!!");

                    break;

            case 2: if((mutex==1)&&(full!=0))

                    consumer();

                    else

                    printf("Buffer is empty!!");

                    break;

            case 3: exit(0);

                    break;

        }

    }

    return 0;

}

int wait(int s)

{

    return (--s);

}

int signal(int s)

{

    return(++s);

}

void producer()

{

    mutex=wait(mutex);

    full=signal(full);

    empty=wait(empty);

    x++;

    printf("\nProducer produces the item %d",x);

    mutex=signal(mutex);

}

void consumer()

{

    mutex=wait(mutex);

    full=wait(full);

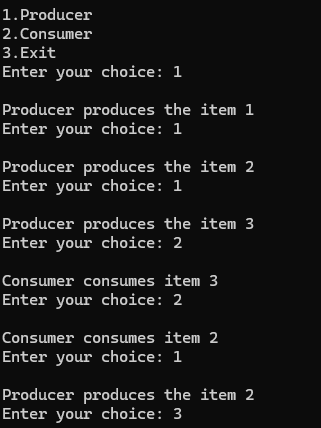
    empty=signal(empty);

    printf("\nConsumer consumes item %d",x);

    x--;

    mutex=signal(mutex);

}



**Write a C program to simulate the concept of Dining-Philosophers problem.**

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#define N 5

#define THINKING 2

#define HUNGRY 1

#define EATING 0

#define LEFT (i + 4) % N

#define RIGHT (i + 1) % N

int state[N];

int phil[N] = {0,1,2,3,4};

sem\_t mutex;

sem\_t S[N];

void test(int i)

{

if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING)

{

state[i] = EATING;

sleep(2);

printf("Philosopher %d takes fork %d and %d\n", i +1, LEFT +1, i +1);

printf("Philosopher %d is Eating\n", i +1);

sem\_post(&S[i]);

}

}

void take\_fork(int i)

{

sem\_wait(&mutex);

state[i] = HUNGRY;

printf("Philosopher %d is Hungry\n",i+1);

test(i);

sem\_post(&mutex);

sem\_wait(&S[i]);

sleep(1);

}

void put\_fork(int i)

{

sem\_wait(&mutex);

state[i] = THINKING;

printf("Philosopher %d putting fork %d and %d down\n",i +1, LEFT +1, i +1);

printf("Philosopher %d is thinking\n", i+1);

test(LEFT);

test(RIGHT);

sem\_post(&mutex);

}

void\* philosopher(void\* num)

{

while (1)

{

int\* i = num;

sleep(1);

take\_fork(\*i);

sleep(0);

put\_fork(\*i);

}

}

int main()

{

int i;

pthread\_t thread\_id[N];

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

for (i =0; i < N; i++)

sem\_init(&S[i],0,0);

for (i =0; i < N; i++)

{

pthread\_create(&thread\_id[i], NULL, philosopher, &phil[i]);

printf("Philosopher %d is thinking\n", i +1);

    }

for (i =0; i < N; i++)

{

pthread\_join(thread\_id[i], NULL);

}

}

