// Task-1:

#include <pthread.h>

#include <stdio.h>

#include <string.h>

#define MAX 10 //producers and consumers can produce and consume upto MAX

#define BUFLEN 6

#define NUMTHREAD 2 /\* number of threads \*/

void \* consumer(int \*id);

void \* producer(int \*id);

char buffer[BUFLEN];

char source[BUFLEN]; //from this array producer will store it's production into buffer

int pCount = 0;

int cCount = 0;

int buflen;

//initializing pthread mutex and condition variables

pthread\_mutex\_t count\_mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t nonEmpty = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_t full = PTHREAD\_COND\_INITIALIZER;

int thread\_id[NUMTHREAD] = {0,1};

int i = 0;

int j = 0;

main()

{

int i;

/\* define the type to be pthread \*/

pthread\_t thread[NUMTHREAD];

strcpy(source,"abcdef");

buflen = strlen(source);

/\* create 2 threads\*/

/\* create one consumer and one producer \*/

/\* define the properties of multi threads for both threads \*/

//Write Code Here

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

{

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

}

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

{

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

}

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

{

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

}

printf("\n");

}

void \* producer(int \*id)

{

/\*

1. Producer stores the values in the buffer (Here copies values from source[] to buffer[]).

2. Use mutex and thread communication (wait(), sleep() etc.) for the critical section.

3. Print which producer is storing which values using which thread inside the critical section.

4. Producer can produce up to MAX

\*/

//Write code here

int i;

pthread\_mutex\_lock(&count\_mutex);

while (pCount == MAX)

{

printf("Producer %d is waiting for empty Thread %d\n", \*id,thread\_id[i]);

pthread\_cond\_wait(&nonEmpty, &count\_mutex);

}

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

{

buffer[i] = source[i];

printf("%d produced %c by Thread %d\n", \*id, source[i],thread\_id[i]);

pthread\_cond\_signal(&full);

pthread\_mutex\_unlock(&count\_mutex);

}

pCount++;

}

void \* consumer(int \*id)

{

/\*

1. Consumer takes out the value from the buffer and makes it empty.

2. Use mutex and thread communication (wait(), sleep() etc.) for critical section

3. Print which consumer is taking which values using which thread inside the critical section.

4. Consumer can consume up to MAX

\*/

//Write code here

int i;

pthread\_mutex\_lock(&count\_mutex);

while (cCount == MAX)

{

printf("Consumer %d is waiting for full Thread %d\n", \*id,thread\_id[i]);

pthread\_cond\_wait(&full, &count\_mutex);

}

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

{

buffer[i] = '\0';

printf("%d consuming %c by Thread %d\n", \*id, source[i],thread\_id[i]);

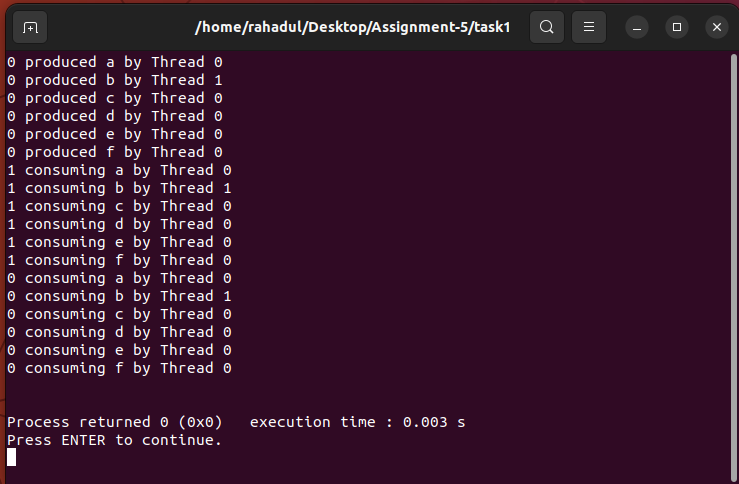
pthread\_cond\_signal(&nonEmpty);

pthread\_mutex\_unlock(&count\_mutex);

}

cCount++;

}



// Task-2:

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

/\*

This program provides a possible solution using mutex and semaphore.

use 5 Farmers and 5 ShopOwners to demonstrate the solution.

\*/

#define MaxCrops 5 // Maximum crops a Farmer can produce or a Shpoowner can take

#define warehouseSize 5 // Size of the warehouse

sem\_t empty;

sem\_t full;

int in = 0;

int out = 0;

char crops[warehouseSize]={'R','W','P','S','M'}; //indicating room for different crops

char warehouse[warehouseSize]={'N','N','N','N','N'}; //initially all the room is empty

pthread\_mutex\_t mutex;

void \*Farmer(void \*far)

{

/\*

1.Farmer harvest crops and put them in particular room. For example, room 0 for Rice(R).

2.use mutex and semaphore for critical section.

3.print which farmer is keeping which crops in which room inside the critical section.

4.print the whole warehouse buffer outside of the critical section

\*/

int i;

int id = (\*(int \*)far);

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

{

sem\_wait(&empty);

pthread\_mutex\_lock(&mutex);

warehouse[in] = crops[id];

in = (in + 1) % warehouseSize;

printf("Farmer %d: Insert crops %c at %d\n", id, crops[id], in);

pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

}

}

void \*ShopOwner(void \*sho)

{

/\*

1.Shop owner takes crops and makes that particular room empty.

2.use mutex and semaphore for the critical section.

3.print which shop owner is taking which crops from which room inside the critical section.

4.print the whole warehouse buffer outside of the critical section

\*/

int i;

int id = \*(int \*)sho;

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

{

sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

out = (out + 1) % warehouseSize;

printf("Shop owner %d: Remove crops %c from %d\n", id, warehouse[out], out);

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty);

}

}

int main()

{

/\*initializing thread,mutex,semaphore

\*/

pthread\_t Far[5],Sho[5];

pthread\_mutex\_init(&mutex, NULL);

sem\_init(&empty,0,warehouseSize);//when the warehouse is full thread will wait

sem\_init(&full,0,0);//when the warehouse is empty thread will wait

int a[5] = {1,2,3,4,5}; //Just used for numbering the Farmer and ShopOwner

/\*create 5 thread for Farmer 5 thread for ShopOwner

-------------------------------------------------

-------------------------------------------------

\*/

int b[5] = {6, 7, 8, 9, 10};

int i;

for (i = 0; i < 5; i++)

{

pthread\_create(&Far[i], NULL, Farmer, &a[i]);

pthread\_create(&Sho[i], NULL, ShopOwner, &b[i]);

}

for (i = 0; i < 5; i++)

{

pthread\_join(Far[i], NULL);

pthread\_join(Sho[i], NULL);

}

// Closing or destroying mutex and semaphore

pthread\_mutex\_destroy(&mutex);

sem\_destroy(&empty);

sem\_destroy(&full);

return 0;

}

