Ονοματεπώνυμο : Νικολάου Έλια

Tαυτότητα : 1012334

Ημερομηνία Παράδοσης : 5/3/2020

TrainProblem.c

/\*\* @file TrainProblem.c

\* @brief This is a program which implemtents Hw\_2 , the main program.

\*

\* The program gets in the command line two parameters m and n

\* , persons and cars number of threads.

\*

\* @see monitor.c

\* @author Elia Nicolaou 1012334

\* @version 1

\* @bug No know bugs.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include "monitor.h"

//number of threads for car / person

static int Car\_threads;

static int Person\_threads;

//procedure for car , calls unload

void \*Car\_procedure(void \*id){

int car\_id = \*(int\*)id;

while(1){

int person\_id = load\_car(car\_id);

unload\_car(car\_id,person\_id);

}

}

//procedure for person, calls take\_ride

void \*Person\_procedure(void \*id){

int person\_id = \*(int\*)id;

take\_ride(person\_id);

printf("- The person with id : %d has complete his ride :)!!\n\n " , person\_id);

pthread\_exit(NULL);

}

//main

int main(int argc, char \* argv[]) {

int i;

//check for parameters

if (argc < 3)

printf("\n Sorry :( Try again by giving the right parametes - N and number of threads");

else {

Person\_threads = atoi(argv[1]);

Car\_threads = atoi(argv[2]);

if(Person\_threads <0 || Car\_threads <0){

printf(" -- We cannot support negative numbers -- " );

exit(1); }

}

printf("\n --- Welcome to TrainProblem --- \n\n\n");

//create person , and car threads

int myid\_person[Person\_threads];

pthread\_t tid\_person[Person\_threads];

int myid\_car[Car\_threads];

pthread\_t tid\_car[Car\_threads];

//create the monitor

create\_monitor(Person\_threads,Car\_threads);

//creating car threads - calling car procedure

printf("- Creating Car threads... - \n\n");

for (i = 0; i < Car\_threads; i++) {

myid\_car[i] = i;

pthread\_create(&tid\_car[i], NULL, &Car\_procedure, (void \*)&myid\_car[i]);

}

//creating person threads - calling person threads

printf("- Creating Person threads... -\n\n");

for (i = 0; i < Person\_threads; i++) {

myid\_person[i] = i;

pthread\_create(&tid\_person[i], NULL, &Person\_procedure, (void \*)&myid\_person[i]);

}

//waiting person threads to complete

printf("- Waiting for all Person threads to be complete... - \n\n");

for (i = 0; i < Person\_threads; i++) {

pthread\_join(tid\_person[i], NULL);

}

printf("\n\n Thank you for using my program! :)\n\n");

return 0;

}

Monitor.c

/\*\* @file monitor.c

\* @brief This is a program which implemtents Hw\_2 , the monitor part.

\*

\* @see qeueue.c

\* @author Elia Nicolaou 1012334

\* @version 1

\* @bug No know bugs.

\*/

#include <pthread.h>

#include <stdlib.h>

#include <stdio.h>

#include "queue.h"

#include "monitor.h"

//variables

static pthread\_once\_t is\_initialized = PTHREAD\_ONCE\_INIT;

static int cars\_num;

static int persons\_num;

static pthread\_mutex\_t mLock;

QUEUE \*passengers;

pthread\_cond\_t \*persons\_cond;

pthread\_cond\_t car\_cond;

void create\_monitor(int persons,int cars){

printf("- Creating monitor - \n\n" );

cars\_num = cars;

persons\_num = persons;

//passengers queue initialization

passengers = (QUEUE\*)malloc(sizeof(QUEUE));

passengers->head = NULL;

passengers->tail = NULL;

passengers->length= 0;

if(passengers == NULL ){

printf("System out of memory...\n");

exit(1); }

//persons c.v initiliazation.

persons\_cond = (pthread\_cond\_t\*)malloc(sizeof(pthread\_cond\_t) \* persons);

if(persons\_cond == NULL ) {

printf("System out of memory...\n");

exit(1); }

int i;

for(i=0 ; i<persons;i++){

pthread\_cond\_init(&persons\_cond[i],NULL);

}

//initialize mutex

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

//initialize c.v car

pthread\_cond\_init(&car\_cond,NULL);

}

//load the passenger to the car.

int load\_car(int car\_id){

//load passenger and remember their id

pthread\_mutex\_lock(&mLock);

//oso to passengers queue en empty , perimene ta cars

while(empty(passengers)){ //an ine diladi 1 (busy)

pthread\_cond\_wait(&car\_cond,&mLock);

}

//epistrefo to proto

int person\_id;

dequeue(passengers,&person\_id);

printf("- Person with id : %d , is in car : %d \n\n", person\_id , car\_id );

pthread\_mutex\_unlock(&mLock);

return person\_id;

}

//take the ride

void take\_ride(int person\_id){

pthread\_mutex\_lock(&mLock);

enqueue(person\_id,passengers);

pthread\_cond\_signal(&car\_cond);

pthread\_cond\_wait(&persons\_cond[person\_id],&mLock);

pthread\_mutex\_unlock(&mLock);

}

//ride completed

void unload\_car(int car\_id, int person\_id){

pthread\_mutex\_lock(&mLock);

pthread\_cond\_signal(&persons\_cond[person\_id]);

printf("- Person with id : %d , has complete his ride with car : %d \n\n", person\_id,car\_id);

pthread\_mutex\_unlock(&mLock);

}

Monitor.h

#ifndef monitor\_h

#define monitor\_h

#include <pthread.h>

#include <stdlib.h>

#include <stdio.h>

#include "queue.h"

void create\_monitor(int persons , int cars);

int load\_car(int car\_id);

void unload\_car(int car\_id, int person\_id);

void take\_ride(int person\_id);

#endif

Queue.c

/\*\* @file monitor.c

\* @brief This is a program which implemtents Hw\_2 , the queue part.

\*

\* @author Elia Nicolaou 1012334

\* @version 1

\* @bug No know bugs.

\*/

#include <stdio.h>

#include <stdlib.h>

#include "queue.h"

int dequeue(QUEUE \*q, int \*retval)

{

    NODE \*p = NULL; // copy pointer used for free()

    if ((q == NULL) || (q->head == NULL)){

     printf("Sorry, queue is empty \n");

     return EXIT\_FAILURE;

    }

    if (retval == NULL){

     printf("Retval is null");

     return EXIT\_FAILURE;

    }

    p = q->head;

    \*retval = q->head->data;

    q->head = q->head->next;

    free(p);

    --(q->length);

    if (q->length == 0){

     q->tail = NULL;

    }

    return EXIT\_SUCCESS;

}

int enqueue(int value, QUEUE \*q)

{

    NODE \*p = NULL;

    if (q == NULL){

     return EXIT\_FAILURE;

    }

    p = (NODE\*)malloc(sizeof(NODE));

    if (p == NULL){

     printf("System out of memory...\n");

     return EXIT\_FAILURE;

    }

    p->data = value;

    p->next = NULL;

    if (q->length == 0)

     q->head = q->tail = p;

    else{

    // append on end

     q->tail->next = p;

     q->tail = p;

    }

     (q->length)++;

    return EXIT\_SUCCESS;

}

int empty(QUEUE \*q){

if(q->length == 0){

return 1;

}

return 0;

}

Queue.h

#ifndef queue\_h

#define queue\_h

typedef struct node{

int data;

struct node \*next;

} NODE;

typedef struct {

NODE \*head;

NODE \*tail;

int length;

}QUEUE;

int empty(QUEUE \*q);

int dequeue(QUEUE \*q, int \*retval);

int enqueue (int value, QUEUE \*q);

#endif