**מעבדה 9 - סמפורים – בעיות נוספות**

**מגישים:** 1. חוטמליאנסקי דמיטרי 334017415

2. שיח אחמד מוחמד 209158120

תרגיל 1

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <semaphore.h>

#include <unistd.h>

#define K 5

#define N 10

void\* tFunc(void\* p);

void stkPush(int num);

int stkPop();

//\*\*\*\*global variables definition\*\*\*\*//

int stk[N];

int idx = 0;

//\*\*\*\*semaphores definition\*\*\*\*//

sem\_t mutex;

sem\_t empty;

sem\_t full;

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

pthread\_t tArr[K];

int i = 0 , ans[K];

//\*\*\*\*\*\*\*semaphores initiatization\*\*\*\*//

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

sem\_init( &empty, 0, 0 );

sem\_init( &full, 0, N );

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

ans[i] = pthread\_create(&tArr[i], NULL, tFunc, NULL);

sleep(10);

return 0;

}

void\* tFunc(void\* p) {

int num;

while(1){

num = rand() % 100;

if(rand() % 2) {

printf("Push %d\n", num);

stkPush(num);

} else {

printf("Pop %d\n", stkPop());

}

sleep(1);

}

}

int stkPop() {

int ret = 0;

sem\_wait(&empty); //wait if the stack is empty

sem\_wait(&mutex); //wait if another thread works on the stack

// and then catch the CPU's time for this code

ret = stk[idx - 1]; //popping

idx--;

sem\_post(&full); //wake-up push thread

sem\_post(&mutex); //end of critical code

return ret;

}

void stkPush(int num) {

sem\_wait(&full); //wait if the stack is full

sem\_wait(&mutex); //wait if another thread works on the stack // and then catch the CPU's time for this code

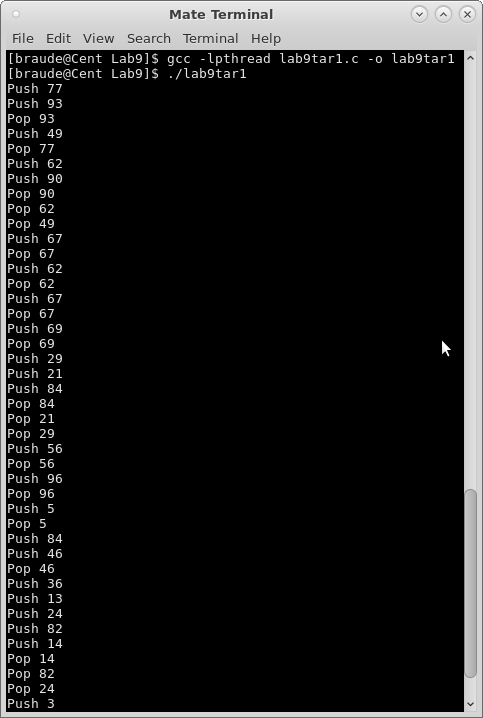
stk[idx] = num; //pushing

idx++;

sem\_post(&empty); //wake-up pop thread

sem\_post(&mutex); //end of critical code

}



תרגיל 2

#include <stdio.h>

#include <semaphore.h>

#include <pthread.h>

#include <unistd.h>

#include <stdlib.h>

#define N 2

#define M 3

//\*\*\*\*semaphores definition\*\*\*//

sem\_t old\_in\_store;

sem\_t young\_in\_store;

sem\_t mutex;

//\*\*\*\*global variable\*\*\*//

int n = 0;

void\* oldManWantsToBuy(void\* ind);

void\* youngManWantsToBuy(void\* ind);

void main(){

pthread\_t t\_Old[N], t\_Young[M]; //threads arrays

int ansOld[N], ansYoung[M], i;

int idOld[N], idYoung[M]; //arrays for saving man id

//\*\*\*\*\*\*semaphores initialization\*\*\*\*//

sem\_init(&old\_in\_store, 0 , 0);

sem\_init(&young\_in\_store, 0, M);

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

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

idOld[i] = i; //save old man id into array

ansOld[i] = pthread\_create(&t\_Old[i], NULL, oldManWantsToBuy, (void\*)&idOld[i]);

//\*\*\*\*\*check if thread creating was success

if(ansOld[i] != 0){

printf("Thread creating error!!!\n");

exit(1);

}

}

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

idYoung[i] = i; //save young man id into array

ansYoung[i] = pthread\_create(&t\_Young[i], NULL, youngManWantsToBuy, (void\*)&idYoung[i]);

//\*\*\*\*\*check if thread creating was success

if(ansYoung[i] != 0){

printf("Thread creating error!!!\n");

exit(1);

}

}

sleep(10);

sem\_destroy(&mutex);

sem\_destroy(&old\_in\_store);

sem\_destroy(&young\_in\_store);

}

void\* oldManWantsToBuy(void\* ind){

int i = \*(int\*)ind;

printf("Old man #%d wants to buy\n", i);

sem\_wait(&mutex); //catch the CPU's time for this code

n = 1; //flag = 1

printf("Old man #%d is buying\n", i);

sleep(1);

printf("Old man #%d is done\n", i);

sem\_post(&mutex); //end of critical code

n = 0; //flag = 0

sem\_post(&old\_in\_store); //signal that the old man left the store

}

void\* youngManWantsToBuy(void\* ind){

int i = \*(int\*)ind;

printf("Young man #%d wants to buy\n", i);

if(n != 0) //if n != 0 signal that old man in the store

sem\_wait(&old\_in\_store); //wait for old man left the store

sem\_post(&young\_in\_store); //one of young man enter into the store

printf("Young man #%d is buying\n", i);

sleep(1);

printf("Young man #%d is done\n", i);

sem\_wait(&young\_in\_store); //signal that the young man left the store

}

