**מעבדה 7 - תזמון תהליכים - תרגיל**

**מגישים:** 1. חוטמליאנסקי דמיטרי 334017415

2. שיח אחמד מוחמד 209158120

תרגיל 1

#include <stdio.h>

#include <pthread.h>

#include <time.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <stdlib.h>

//\*\*\*\*\*function declarations\*\*\*\*\*\*\*

void \*f(void\* thread\_num);

//\*\*\*\*\*global values\*\*\*\*\*\*\*\*\*\*

int this\_thread;

int N;

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

int i,j;

int \*ans, \*t\_num; //pointers to arrays

pthread\_t\* t\_array;

if(argc != 2){ //check if has only 1 argument

printf("\nError parameters");

exit(0);

}

if((N = atoi(argv[1])) == 0){ //check correct input

printf("\nIllegal number of threads");

exit(0);

}

//\*\*\*\*\*\*\*\*\*\*memory allocation for arrays\*\*\*\*\*\*\*\*\*\*

if((ans = (int\*)malloc(N\*sizeof(int))) == NULL){

printf("\nError memory allocation of 'ANS'");

exit(0);

}

if((t\_num = (int\*)malloc(N\*sizeof(int))) == NULL){

printf("\nError memory allocation of 'T\_NUM'");

exit(0);

}

if((t\_array = (pthread\_t\*)malloc(N\*sizeof(pthread\_t))) == NULL){

printf("\nError memory allocation of 'T\_ARRAY'");

exit(0);

}

//\*\*\*\*\*\*\*\*\*\*creating N threads\*\*\*\*\*\*\*\*\*\*\*\*

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

t\_num[i] = i+1;

ans[i] = pthread\_create(&t\_array[i], NULL, f, (void\*)&t\_num[i]);

if(ans[i]){ //check if creating was successful

printf("\nCan't create thread #%d", i);

exit(0);

}

}

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

pthread\_join(t\_array[i], NULL); //wait for threads ending

}

//\*\*\*\*\*\*\*\*\*\*freeing allocated memory\*\*\*\*\*\*

free(ans);

free(t\_num);

free(t\_array);

return 0;

}

void \*f(void\* thread\_num){

double time;

struct timeval t1, t2;

while(1){

this\_thread = \*(int\*)thread\_num;

gettimeofday(&t1, NULL);

//\*\*\*\*\*\*\*run in the while loop untill it's the same thread

while(this\_thread == \*(int\*)thread\_num){

gettimeofday(&t2, NULL);

}

//convert to milliseconds

time = (t2.tv\_sec - t1.tv\_sec) \* 1000.0;

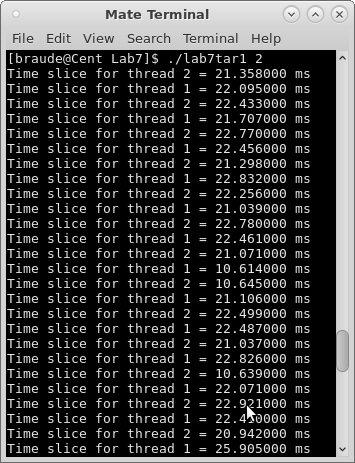
time += (t2.tv\_usec - t1.tv\_usec) / 1000.0;

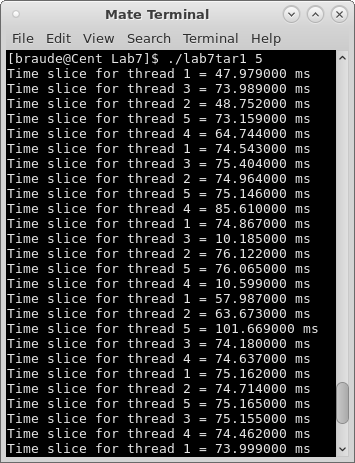
printf("Time slice for thread %d = %0.6f ms\n", this\_thread, time);

this\_thread = \*(int\*)thread\_num;

}

}





תרגיל 2

#include <stdio.h>

#include <pthread.h>

#include <time.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <stdlib.h>

//\*\*\*\*\*function declarations\*\*\*\*\*\*\*

void \*f(void\* thread\_num);

void \*print\_count(void \*arg);

//\*\*\*\*\*global values\*\*\*\*\*\*\*\*\*\*

int this\_thread;

int N;

int sec;

int\* count\_array;

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

int i,j;

int \*ans, \*t\_num; //pointers to arrays

pthread\_t\* t\_array;

if(argc != 2){ //check if has only 1 argument

printf("\nError parameters");

exit(0);

}

if((N = atoi(argv[1])) == 0){ //check correct input

printf("\nIllegal number of threads");

exit(0);

}

//\*\*\*\*\*\*\*\*\*\*memory allocation for arrays\*\*\*\*\*\*\*\*\*\*

if((ans = (int\*)malloc((N + 1) \* sizeof(int))) == NULL){

printf("\nError memory allocation of 'ANS'");

exit(0);

}

if((t\_num = (int\*)malloc(N \* sizeof(int))) == NULL){

printf("\nError memory allocation of 'T\_NUM'");

exit(0);

}

if((t\_array = (pthread\_t\*)malloc((N + 1) \* sizeof(pthread\_t))) == NULL){

printf("\nError memory allocation of 'T\_ARRAY'");

exit(0);

}

if((count\_array=(int\*)malloc(N \* sizeof(int))) == NULL){

printf("\nError memory allocation of 'COUNT\_ARRAY'");

exit(0);

}

//\*\*\*\*\*\*\*\*\*\*set all count\_array's cells to (0)\*\*\*\*\*\*\*\*\*\*\*\*

for (i = 0; i < N + 1; i++){

count\_array[i]=0;

}

//\*\*\*\*\*\*\*\*\*\*creating N threads\*\*\*\*\*\*\*\*\*\*\*\*

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

t\_num[i] = i;

ans[i] = pthread\_create(&t\_array[i], NULL, f, (void\*)&t\_num[i]);

if(ans[i]){ //check if creating was successful

printf("\nCan't create thread #%d", i);

exit(0);

}

}

//\*\*\*\*\*\*\*\*\*\*one more thread for printing\*\*\*\*\*\*

ans[N] = pthread\_create(&t\_array[N], NULL, print\_count, NULL);

for(i = 0; i < N + 1; i++){

pthread\_join(t\_array[i], NULL); //wait for threads ending

}

//\*\*\*\*\*\*\*\*\*\*freeing allocated memory\*\*\*\*\*\*

free(ans);

free(t\_num);

free(t\_array);

free(count\_array);

return 0;

}

//\*\*\*\*\*\*\*\*\*printing counters of threads function\*\*\*\*\*\*\*\*

void \*print\_count(void \*arg){

int i;

sec = 0;

while(sec <= 20){

printf("Past %d seconds\n", sec);

sec += 2;

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

printf("Thread #%d -> %d\n", i + 1, count\_array[i]);

}

printf("\n");

sleep(2);

}

}

//\*\*\*\*\*\*\*\*\*running threads function\*\*\*\*\*\*\*\*\*\*

void \*f(void\* thread\_num){

while(sec <= 20){

this\_thread = \*(int \*)thread\_num;

count\_array[this\_thread]++;

while(this\_thread == \*(int\*)thread\_num){

}

this\_thread = \*(int\*)thread\_num;

}

}

