Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 3 по курсу**

**«Операционные системы»**

**Управление потоками в ОС**

Студент: Почечура Артемий Андреевич

Группа: М8О–206Б–20

Вариант: 13

Преподаватель: Соколов Андрей Алексеевич

Оценка: 5

Дата: 13.11.21

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Постановка задачи**

## Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 13:

Есть набор 128 битных чисел, записанных в шестнадцатеричном представлении, хранящихся в файле. Необходимо посчитать их среднее арифметическое. Округлить результат до целых. Количество используемой оперативной памяти должно задаваться "ключом"

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы с потоками в C.
2. Написать вспомогательные функции для работы с матрицами.
3. С помощью принципа разбиения задачи на маленькие подзадачи, реализовать функцию, которая будет проводить частичную обработку данных, для ее запуска в многопоточном режиме.
4. Реализовать функцию, которая будет создавать потоки, запускать потоковую функцию, управлять потоками.
5. Реализовать обработку системных ошибок согласно заданию

**Основные файлы программы**

**main.c:**

#include <pthread.h>

#include <stdlib.h>

#include <mcheck.h>

#include <stdio.h>

#include <stdbool.h>

#include <stdint.h>

#include <inttypes.h>

pthread\_mutex\_t mutex;

typedef struct{

\_\_uint128\_t value;

int cnt;

FILE\* f;

} pthrData;

bool status=false;

#define memory 692

#define thread\_size 12 //704-692

void\* threadFunc(void\* thread\_data){

pthrData \*data = (pthrData\*) thread\_data;

while(true){

if(status){

break;

}

pthread\_mutex\_lock(&mutex);

char s[33];

int k=0;

if(fscanf(data->f,"%c",&s[k])==-1){

break;

}

while(s[k]==' ' || s[k]=='\n' || s[k]=='\t'){

if(fscanf(data->f,"%c",&s[k])==-1){

status=true;

break;

}

}

if(status){

break;

}

while(s[k]!=' ' && s[k]!='\n' && s[k]!='\t'){

k++;

if(k>31){

return (void\*)-1;

}

if(fscanf(data->f,"%c",&s[k])==-1){

status=true;

break;

}

}

\_\_uint128\_t a=0;

for(int i=0;i<k;i++){

if(s[i]>='0' && s[i]<='9'){

a=a\*16+(s[i]-'0');

}else if(s[i]>='A' && s[i]<='F'){

a=a\*16+(s[i]-'A'+10);

} else {

return (void\*)-1;

}

}

/\*while(true){

}\*/

data->value=data->value+a;

data->cnt++;

if(status){

break;

}

pthread\_mutex\_unlock(&mutex);

}

pthread\_mutex\_unlock(&mutex);

return NULL;

}

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

int n=1;

if(argc!=2 && argc!=3){

fprintf(stderr, "Неверное количество аргументов\n");

return 1;

}

if(argc==2){

n=1;

} else {

long long p=atoi(argv[2]);

if(p<1){

fprintf(stderr, "Количество памяти не может быть меньше единицы\n");

return 2;

}

p=p-memory;

if(p<0){

fprintf(stderr, "Количество памяти слишком мало\n");

return 3;

}

n=n+p/thread\_size;

}

\_\_uint128\_t sum=0;

int count=0;

FILE \*in = fopen(argv[1], "r");

if(in==NULL){

fprintf(stderr, "Что-то не так с файлом '%s'\n", argv[1]);

return 3;

}

pthread\_t threads[n];

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

pthrData threadData[n];

for(int i=0;i<n;i++){

threadData[i].value=0;

threadData[i].cnt=0;

threadData[i].f=in;

pthread\_create(&(threads[i]), NULL, threadFunc, &threadData[i]);

}

for(int i=0;i<n;i++){

void\* r;

pthread\_join(threads[i], &r);

if(r==(void\*)-1){

fprintf(stderr, "Неправильные данные\n");

return 4;

}

}

for(int i=0;i<n;i++){

sum=sum+threadData[i].value;

count=count+threadData[i].cnt;

}

sum=sum/count;

long long g=(sum>>64);

long long q=sum&0xFFFFFFFFFFFFFFFF;

if(g>0){

printf("%llx\n",g);

}

printf("%llx\n",q);

pthread\_mutex\_destroy(&mutex);

}

**test:**

3E8

7D0

BB8

**Пример работы**

root@DESKTOP-5HM2HTK:~# ./a.out

Неверное количество аргументов

root@DESKTOP-5HM2HTK:~# ./a.out test 500

Количество памяти слишком мало

root@DESKTOP-5HM2HTK:~# ./a.out test 700

7d0

root@DESKTOP-5HM2HTK:~# ./a.out test

7d0

**Вывод**

В ходе выполнения данной лабораторной работы я познакомился с тем, как производится работа с потоками в языке Си. Основная идея применения потоков состоит в разбиение большой задачи на множество более мелких задач, которые выполняются параллельно. Но данный подход требует особенной внимательности программиста к совместно используемым данным и тщательного продумывания алгоритма.