Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу
«Операционные системы»

Студент: Скрипачев Фёдор Михайлович
Группа: М8О-209Б-23
Вариант: 16
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2025

Содержание

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

Репозиторий

https://github.com/gthcbr25/osi/tree/main/oslab2

Постановка задачи

Цель работы

Приобретение практических навыков в:
· Управление потоками в ОС
· Обеспечение синхронизации между потоками

Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска программы.
Необходимо уметь продемонстрировать количество потоков, используемых программой, с помощью стандартных средств операционной системы.
Привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Объяснить получившиеся результаты.
Вариант 16: задаётся радиус окружности. Необходимо с помощью метода Монте-Карло рассчитать её площадь

Общие сведения о программе

Программа написана на языке Си в UNIX-подобной операционной системе. Для компиляции программы требуется указать ключ -pthread. Для запуска программы в качестве 1 аргумента командной строки необходимо указать количество потоков, в качестве 2 аргумента – радиус окружности.

Общий метод и алгоритм решения

На вход от пользователя поступает количество потоков и радиус окружности. Метод Монте-Карло построен на том, что можно выбрать n случайных точек внутри квадрата со стороной 2R и по количеству попавших точек внутрь круга получить площадь круга. Каждому потоку выделяется равное количество точек, которое они должны сгенерировать и проверить входят ли они внутрь круга. Я решил использовать единую структуру, с которой будут взаимодействовать потоки и

увеличивать счетчик внутри неё при генерации подходящей точки. Далее зная исходное число точек и площадь квадрата с помощью метода Монте-Карло программа вычислит и выведет площадь круга.

Исходный код

main.c

```
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <semaphore.h>



typedef struct {
    int N;
    unsigned int max_threads;
    float r;
    int m;
    pthread_mutex_t* mutex;

}Data;



void *calculate(void* arg){
    printf("Thread ID: %lu\n", pthread_self());
    Data* data = (Data*)arg;
    int c = 0;
    int count = data->N / data->max_threads;
    for (int i = 0; i < count; i++) {
        float x = ((float)rand()/(float)(RAND_MAX) * data->r);
```

```
        float y = ((float)rand()/(float)(RAND_MAX) * data->r);
        if (rand() % 2 == 1) x *= -1;
        if (rand() % 2 == 1) y *= -1;
        if (x * x + y * y <= data->r * data->r) {
            c++;
        }
    }
    pthread_mutex_lock(data->mutex);
    data->m += c;
    pthread_mutex_unlock(data->mutex);
    pthread_exit(0);
}



int main(int argc, char *argv[]){
    if (argc < 2){
        fprintf(stderr,"Usage %s <max threads> <radius>", argv[0]);
        return 1;
    }

    float r = atof(argv[2]);
    unsigned int max_threads = atoi(argv[1]);



    pthread_mutex_t mutex;
    pthread_mutex_init(&mutex, NULL);

    int m = 0;
```

```
int N = (int) 1000 * r * r;
srand(time(NULL));

pthread_t* threads = malloc(max_threads * sizeof(pthread_t));

if (!threads){
    fprintf(stderr, "Failed to allocated memory");
    return 1;
}

Data data = {N, max_threads, r, m, &mutex};

clock_t start = clock();

for (int i = 0; i <max_threads; ++i){
    if(pthread_create(&threads[i], NULL, calculate, &data) != 0){
        fprintf(stderr, "Failed create thread");
        return 1;
    }
}

for (int i = 0; i < max_threads; ++i){
    pthread_join(threads[i], NULL);
}

for (int i = 0; i < N % max_threads; i++) {
    float x = ((float)rand()/(float)(RAND_MAX) * r) * (-1 * (rand() % 2));
    float y = ((float)rand()/(float)(RAND_MAX) * r) * (-1 * (rand() % 2));
```

```
        if (x * x + y * y <= r * r) {
            data.m++;
        }
    }
    clock_t end = clock();
    double del_time = (double) end-start;
    printf("Square: %f\n", ((float)data.m / (float)N) * 4 * r * r);
    printf("Del time: %f\n", del_time);

    free(threads);
    pthread_mutex_destroy(&mutex);


    return 0;
}
```

Демонстрация работы программы

```
gcc main.c
./a.out 3 9
Thread ID: 140209670846144
Thread ID: 140209662453440
Thread ID: 140209654060736
Square: 254.891998 Del time: 50314.000000
```

Выводы

В языке программирования C многопоточность реализуется с помощью библиотеки pthread.h, которая позволяет работать с потоками операционной системы. Создание потоков происходит быстрее, чем создание процессов, поскольку потоки разделяют общую память в рамках одного процесса. Это делает многопоточность эффективным способом ускорения обработки данных: задачи, которые не зависят друг от друга и имеют однотипный характер, можно распределить между несколькими потоками, выполняющимися параллельно.