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

Лабораторная работа №3 по курсу
«Операционные системы»

Студент: Денисов Максим
Группа: М8О-207Б-21
Вариант: 16
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/ClownOff/OS

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

Для компиляции программы требуется указать ключ -pthread. Для запуска программы в качестве 1 аргумента командной строки необходимо указать радиус окружности, в качестве 2 аргумента - количество проверяемых точек, в качестве 3 аргумента - количество потоков.

- pthread_create() — создание потока с передачей ему аргументов. В случае успеха возвращает 0.
- pthread_join() — ожидает завершения потока обозначенного THREAD_ID. Если этот поток к тому времени был уже завершен, то функция немедленно возвращает значение.

Общий метод и алгоритм решения

Пусть на вход от пользователя поступило n точек и m потоков. Тогда каждый поток будет обрабатывать n/m точек. Чтобы избежать работы с критической областью памяти будем хранить количество точек, попавших в круг и обрабатываемых i-м потоком, в динамическом массиве N.

Каждый поток обрабатывает n/m точек. Для этого генерируется случайная точка с координатами x, y, лежащая в пределах квадрата $(-r \leqslant x \leqslant r, -r \leqslant y \leqslant r)$. При этом если точка лежит в пределах круга (удовлетворяет неравенству $x^2 + y^2 \leqslant R^2$) , то инкрементируется значение N[i]. В качестве аргумента поток принимает количество обрабатываемых точек и номер i. После завершения работы всех потоков выводится площадь, вычисленная методом Монте-Карло и стандартным методом

Исходный код

```
#include <stdio.h>

#include <math.h>

#include <pthread.h>

#include <stdlib.h>

#include <time.h>



int R;

int *N;

typedef struct arguments {

    int points;

    int i;

} Arg;



double get_rand() { // возврат рандомного числа от 0 до 1

    return ((double) rand()) / RAND_MAX;
```

```
}


double get_rand_range(double min, double max) { // возвращает рандомное число от min до max
    return get_rand() * (max - min) + min;
}


void *thread_function(void *args) { // ссоздаёт n рандомных точек в квадрате размером 2*R и проверяет находятся ли точки в круге
    Arg *arg = (Arg *) args;
    int n = arg->points;
    int i = arg->i;
    for (int j = 0; j < n; j++) {
        double x = get_rand_range(-R, R);
        double y = get_rand_range(-R, R);
        if (x * x + y * y <= R * R) {
            N[i]++;
        }
    }
    return NULL;
}


int main(int argc, char *argv[]) {
    if (argc != 4) {
        printf("./a.out Radius Number_of_points Number_of_threads\n");
        exit(1);
```

```
    }


    R = atoi(argv[1]);
    int points_num = atoi(argv[2]), threads_num = atoi(argv[3]);
    N = (int *) calloc(threads_num, sizeof(int)); // массив точек в круге
    double time_spent = 0.0;
    clock_t begin = clock();
    pthread_t *threads = (pthread_t *) calloc(threads_num, sizeof(pthread_t));
    if (threads == NULL) {
        printf("Can't allocate memory for threads\n");
        exit(1);
    }
    int points_for_thread = points_num / threads_num;
    Arg a;
    for (int i = 0; i < threads_num; i++) {
        a.points = points_for_thread + (i < (points_num % threads_num));
        a.i = i;
        if (pthread_create(&threads[i], NULL, thread_function, &a) != 0) {
            printf("Can not create thread\n");
            exit(1);
        }


    }
    for (int i = 0; i < threads_num; i++) {
        if (pthread_join(threads[i], NULL) != 0) {
```

```
            printf("Join error\n");
            exit(1);
        }

    }
    double n = 0;
    for (int i = 0; i < threads_num; i++) { // подсчёт точек в круге
        n += N[i] / 1.0 / points_num;
    }
    printf("Monte-Carlo Circle square is %.5f\n",
           (double) 4 * R * R * n);
    printf("Real Circle square is %.5f\n", (double) M_PI * R * R);
    clock_t end = clock();
    time_spent += (double)(end - begin);
    printf("The elapsed time is %f seconds\n", time_spent),
    free(threads);
    return 0;
}
```

Демонстрация работы программы

Выводы

Язык Си позволяет пользователю взаимодействовать с потоками операционной системы. Для этого на Unix-подобных системах требуется подключить библиотеку pthread.h.

МЫногопоточность – один из способов ускорить обработку каких-либо данных: выполнение однотипных, не зависящих друг от друга задач, можно поручить отдельным потокам, которые будут работать параллельно.

Средствами языка Си можно совершать системные запросы на создание потока, ожидания завершения потока, а также использовать различные примитивы синхронизации.