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: Примаченко Александр Александрович
Группа: М8О-208Б-20
Вариант: 14
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/SashaPaladin/OS/tree/main/3_lab

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 14: есть колода из 52 карт, рассчитать экспериментально (метод Монте-Карло) вероятность того, что сверху лежат две одинаковых карты. Количество раундов подается с ключом.

Общие сведения о программе

Программа написана на языке Си в UNIX-подобной операционной системе (Fedora 34). 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.

Исходный код

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

int *N;
typedef struct arguments {
   int points;
   int i;
} Arg;

double get_rand() { // return random double from 0 to 1
   return ((double) rand()) / RAND_MAX;
}

double get_rand_range(double min, double max) { // return random double from min to max
   return get_rand() * (max - min) + min;
}

void *thread_function(void *args) { // create n random points and cheak
   Arg *arg = (Arg *) args;
   int n = arg->points;
   int i = arg->i;
   for (int j = 0; j < n; j++) {
      double x = get_rand_range(0, 52);
      double y = get_rand_range(0, 51);
      for (int k = 0; k < 52; k += 4) {
         if (k <= x && x < k + 4 && k <= y && y < k + 3) N[i]++;
      }
   }
   return NULL;
}

int main(int argc, char *argv[]) {
   if (argc != 3) {
      printf("Syntax: ./*executable_file_name* Rounds Number_of_threads\n");
      exit(1);
   }
   int Rounds = atoi(argv[1]), threads_num = atoi(argv[2]);
   N = (int *) calloc(threads_num, sizeof(int)); // array of number points
   srand(time(NULL));
   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 = Rounds / threads_num;
   Arg a;
   for (int i = 0; i < threads_num; i++) {
      a.points = points_for_thread + (i < Rounds % 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++) { // calculate points
        n += (double) N[i] / Rounds;
    }
    printf("Monte-Carlo chance is %.5f\n", (double) n);
    printf("Real chance is %.5f\n", (double) 1 / 17);
    free(threads);
    return 0;
}
```

Выводы

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

Создание потоков происходит быстрее, чем создание процессов, а все потоки используют одну и ту же область данных. Поэтому многопоточность – 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.