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: Леу Иван Александрович

Группа: М8О-209Б-23

Вариант: 9

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Содержание**

* Репозиторий
* Постановка задачи
* Общие сведения о программе
* Общий метод и алгоритм решения
* Исходный код
* Демонстрация работы программы
* Выводы

**Репозиторий**

[**https://github.com/IvanLeu/OS\_Labs/tree/main/lab2**](https://github.com/IvanLeu/OS_Labs/tree/main/lab2)

**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 9: Рассчитать детерминант матрицы (используя определение детерминанта)

**Общие сведения о программе**

Программа написана на языке Си в операционной системе Windows. Для запуска программы в качестве аргумента командной строки необходимо указать максимальное количество используемых потоков.

**Общий метод и алгоритм решения**

Пусть на вход от пользователя поступило t потоков и матрица размеров n x m. Тогда каждый поток будет считать детерминант рекурсино, по определению детерминанта. Каждый поток будет вызывать функцию подсчета детерминанта матрицы, на порядок ниже. Рекурсия повторяется пока порядок матрицы не равен 1. Результаты работы потоков щаписываются в массив и суммируются.

**Исходный код**

main.cpp

#include <Windows.h>

#include <iostream>

#include <vector>

struct ThreadData {

std::vector<std::vector<int>> matrix;

size\_t size;

size\_t row;

size\_t column;

int nThreads;

HANDLE semaphore;

};

int calculateDeterminant(const std::vector<std::vector<int>>& matrix, size\_t size, size\_t nThreads);

DWORD CALLBACK calculateSubDeterminant(LPVOID param) {

ThreadData\* data = static\_cast<ThreadData\*>(param);

size\_t newSize = data->size - 1;

std::vector<std::vector<int>> subMatrix(newSize, std::vector<int>(newSize));

for (int i = 1; i < data->size; i++) {

int colIndex = 0;

for (int j = 0; j < data->size; ++j) {

if (j == data->column) {

continue;

}

subMatrix[i - 1][colIndex] = data->matrix[i][j];

colIndex++;

}

}

int result = calculateDeterminant(subMatrix, newSize, data->nThreads);

ReleaseSemaphore(data->semaphore, 1, NULL);

return result;

}

int calculateDeterminant(const std::vector<std::vector<int>>& matrix, size\_t size, size\_t nThreads) {

if (size == 1) {

return matrix[0][0];

}

HANDLE semaphore = CreateSemaphore(NULL, nThreads, nThreads, NULL);

int determinant = 0;

HANDLE\* threads = new HANDLE[size];

ThreadData\* data = new ThreadData[size];

int\* results = new int[size];

for (int col = 0; col < size; ++col) {

WaitForSingleObject(semaphore, INFINITE);

data[col].matrix = matrix;

data[col].size = size;

data[col].row = 0;

data[col].column = col;

data[col].semaphore = semaphore;

data[col].nThreads = nThreads;

threads[col] = CreateThread(NULL, 0, calculateSubDeterminant, &data[col], 0, NULL);

}

WaitForMultipleObjects(size, threads, TRUE, INFINITE);

for (int col = 0; col < size; ++col) {

DWORD code;

GetExitCodeThread(threads[col], &code);

results[col] = code;

determinant += ((col % 2 == 0 ? 1 : -1) \* matrix[0][col] \* results[col]);

CloseHandle(threads[col]);

}

delete[] threads;

delete[] data;

delete[] results;

CloseHandle(semaphore);

return determinant;

}

int main() {

size\_t maxThreads;

std::cout << "Enter max amount of threads: ";

std::cin >> maxThreads;

size\_t size;

std::cout << "Enter matrix size: ";

std::cin >> size;

std::vector<std::vector<int>> matrix(size, std::vector<int>(size));

std::cout << "Enter matrix elements: ";

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

for (int j = 0; j < size; ++j) {

std::cin >> matrix[i][j];

}

}

int determinant = calculateDeterminant(matrix, size, maxThreads);

std::cout << determinant << std::endl;

return 0;

}

**Демонстрация работы программы**

**PS C:\Users\ivanleu\Documents\C++\os\_labs\2\src>** cd .\build\

**PS C:\Users\ivanleu\Documents\C++\os\_labs\2\src\build>** ./main 1

33458.84ms

**PS C:\Users\ivanleu\Documents\C++\os\_labs\2\src\build>** ./main 2

31183.74ms

**PS C:\Users\ivanleu\Documents\C++\os\_labs\2\src\build>** ./main 5

30220.99ms

**PS C:\Users\ivanleu\Documents\C++\os\_labs\2\src\build>** ./main 10

29515.58ms

**Выводы**

Язык Си позволяет пользователю взаимодействовать с потоками операционной системы. Для этого при работе на системе Windows требуется подключить библиотеку <windows.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.