Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Управление потоками в ОС**

Студент: Епифанов Евгений Валерьевич

Группа: М8О–212Б–22

Вариант: 10

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

## Цель работы

## Целью является приобретение практических навыков в: • Управление потоками в ОС • Обеспечение синхронизации между потоками

## Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, 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 входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 10: Решить систему линейных уравнений методом Гауса.

**Общие сведения о программе**

Программа компилируется из файла main.cpp. В программе используются следующие системные вызовы:

1. **pthread\_create** - создание нового потока.
2. **pthread\_join** - присоединение к основному потоку после завершения работы.
3. **pthread\_exit** - завершение работы потока с указанным кодом.

Программа решает систему из линейных уравнений, количество которых задается пользователем. Количество переменных m в уравнениях также задается пользователем. Матрица с уравнениями представляется в виде массива длинной (m+1) \* n. При вызове требуемой функции solve\_system в цикле выбираются два индекса: на первые элементы строк, из которой мы будем вычитать и которую мы будем вычитать. Данные процессы разбиваются на заранее заданное количество потоков и выполняются в отдельной функции to\_diagonal. После завершения всех процессов вычитания строк пользователь получает матрицу диагонального вида, представленную в виде одномерного массива. Далее по индексам выводятся элементы, в которых содержатся переменные, являющиеся решениями данной системы.

**Общий метод и алгоритм решения**.

Для реализации задачи необходимо:

1. Написать функцию to\_diagonal.
2. Написать функцию solve\_system и связать её с to\_diagonal.
3. Реализовать чтение ключа, задающего кол-во потоков.
4. Адаптировать функции to\_diagonal и solve\_system под работу с потоками.
5. Проанализировать зависимость времени работы программы от количества используемых потоков.

**Основные файлы программы**

**main.cpp:**

#include <iostream>

#include <vector>

#include <string>

#include <pthread.h>

typedef struct

{

double \* vec;

size\_t sub\_from;

size\_t what\_sub;

size\_t length;

} vec\_with\_ind;

void \* to\_diagonal(void \* data)

{

double \* equation = ((vec\_with\_ind \*)data)->vec;

size\_t sub\_from = ((vec\_with\_ind \*)data)->sub\_from;

size\_t what\_sub = ((vec\_with\_ind \*)data)->what\_sub;

size\_t length = ((vec\_with\_ind \*)data)->length;

double mult = equation[sub\_from + (what\_sub / length)] / equation[what\_sub + (what\_sub / length)];

//std::cout << sub\_from << " " << what\_sub << std::endl;

for (size\_t i = sub\_from; i < sub\_from + length; ++i)

{

//std::cout << sub\_from << ") " << equation[i] << std::endl;

equation[i] = equation[i] - equation[what\_sub + i - sub\_from] \* mult;

//std::cout << sub\_from << ") " << equation[i] << std::endl;

}

pthread\_exit(0);

}

void solve\_system(double \* vec, size\_t n, size\_t m, int threads)

{

pthread\_t tid[threads];

for (size\_t i = 0; i < n\*(m+1); i = i + m + 1)

{

for (size\_t j = 0; j < n\*(m+1); j = j + m + 1)

{

int created = 0;

while ((created != threads) and (j < n\*(m+1)))

{

if (i == j)

{

j = j + m + 1;

continue;

}

vec\_with\_ind \* data = (vec\_with\_ind \*)malloc(sizeof(vec\_with\_ind));

data->vec = vec;

data->sub\_from = j;

data->what\_sub = i;

data->length = m + 1;

pthread\_create(&tid[created], NULL, to\_diagonal, data);

created++;

j = j + m + 1;

}

j = j - m - 1;

for (size\_t k = 0; k < threads; ++k) pthread\_join(tid[k], NULL);

}

}

}

void print\_system (double \* vec, size\_t n, size\_t m)

{

for (size\_t i = 0; i < n\*(m+1); ++i)

{

if ((i+1) % (m+1) == 0 and i != 0)

std::cout << "| ";

std::cout << vec[i] << " ";

if ((i+1) % (m+1) == 0 and i != 0)

std::cout << std::endl;

}

std::cout << std::endl;

}

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

{

if (argc != 2)

{

std::cerr << "Incorrect arguments." << std::endl;

return 1;

}

int threads = atoi(argv[1]);

std::cout << "Enter the number of equations of the system:" << std::endl;

int n\_of\_eq;

std::cin >> n\_of\_eq;

std::cout << "Enter the number of variables in the equations:" << std::endl;

int n\_of\_var;

std::cin >> n\_of\_var;

if (n\_of\_eq <= 0 or n\_of\_var <= 0)

{

std::cerr << "Incorrect parameters." << std::endl;

return 1;

}

if (n\_of\_eq != n\_of\_var)

{

std::cerr << "There may be no or an infinite number of solutions." << std::endl;

return 1;

}

std::cout << "Enter the equations: " << std::endl;

double s\_of\_eq[n\_of\_eq \* (n\_of\_var + 1)];

size\_t zero\_cnt = 0;

for (size\_t i = 0; i < n\_of\_eq \* (n\_of\_var + 1); ++i)

{

std::cin >> s\_of\_eq[i];

if (s\_of\_eq[i] == 0) zero\_cnt++;

if (((i + 1) % (n\_of\_var + 1) == 0) and (i != 0))

{

if ((zero\_cnt == n\_of\_var) and (s\_of\_eq[i] != 0))

{

std::cerr << "There are no solutions." << std::endl;

return 1;

}

zero\_cnt = 0;

}

}

std::cout << std::endl;

std::cout << "Your system of equations: " << std::endl;

print\_system(s\_of\_eq, n\_of\_eq, n\_of\_var);

auto start = std::chrono::high\_resolution\_clock::now();

solve\_system(s\_of\_eq, n\_of\_eq, n\_of\_var, threads);

std::cout << "The system of equations in diagonal form: " << std::endl;

print\_system(s\_of\_eq, n\_of\_eq, n\_of\_var);

size\_t ind = 1;

for (size\_t i = 0; i < n\_of\_eq \* (n\_of\_var+1); i = i + n\_of\_var + 2)

{

std::cout << "x" << ind << ") " << s\_of\_eq[i + (n\_of\_var - (i / (n\_of\_var+1)))] / s\_of\_eq[i] << std::endl;

ind++;

}

auto end = std::chrono::high\_resolution\_clock::now();

auto time = std::chrono::duration\_cast<std::chrono::microseconds>(end - start);

std::cout << "The system of equations is solved in " << time.count() << " microseconds." << std::endl;

}

**CMakeLists.txt:**

cmake\_minimum\_required(VERSION 3.10)

project(lab2)

set(CMAKE\_CXX\_STANDARD 20)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

add\_executable(main main.cpp)

**Пример работы**

anatolii@MacBook-Pro-Anatolii ~/D/O/E/o/s/build (main)> ./main 2  
Enter the number of equations of the system:  
5  
Enter the number of variables in the equations:  
5  
Enter the equations:  
1 1 1 1 0 5  
0 1 1 1 1 1  
1 0 1 1 1 2  
1 1 0 1 1 0  
1 1 1 0 1 4  
  
Your system of equations:  
1 1 1 1 0 | 5  
0 1 1 1 1 | 1  
1 0 1 1 1 | 2  
1 1 0 1 1 | 0  
1 1 1 0 1 | 4  
  
The system of equations in diagonal form:  
1 0 0 0 0 | 2  
0 1 0 0 0 | 1  
0 0 1 0 0 | 3  
0 0 0 1 0 | -1  
0 0 0 0 4 | -8  
  
x1) 2  
x2) 1  
x3) 3  
x4) -1  
x5) -2  
The system of equations is solved in 1512 microseconds.

anatolii@MacBook-Pro-Anatolii ~/D/O/E/o/s/build (main) [1]> /Users/anatolii/Desktop/Oper\_Syst/Eugen/os\_lab\_2/src/build/main 10  
Enter the number of equations of the system:  
5  
Enter the number of variables in the equations:  
5  
Enter the equations:  
1 1 1 1 0 5  
0 1 1 1 1 1  
1 0 1 1 1 2  
1 1 0 1 1 0  
1 1 1 0 1 4  
  
Your system of equations:  
1 1 1 1 0 | 5  
0 1 1 1 1 | 1  
1 0 1 1 1 | 2  
1 1 0 1 1 | 0  
1 1 1 0 1 | 4  
  
The system of equations in diagonal form:  
1 0 0 0 0 | 2  
0 1 0 0 0 | 1  
0 0 1 0 0 | 3  
0 0 0 1 0 | -1  
0 0 0 0 4 | -8  
  
x1) 2  
x2) 1  
x3) 3  
x4) -1  
x5) -2  
The system of equations is solved in 961 microseconds.

anatolii@MacBook-Pro-Anatolii ~/D/O/E/o/s/build (main)> ./main 10  
Enter the number of equations of the system:  
2  
Enter the number of variables in the equations:  
2  
Enter the equations:  
1 2 3  
2 3 4  
  
Your system of equations:  
1 2 | 3  
2 3 | 4  
  
The system of equations in diagonal form:  
1 0 | -1  
0 -1 | -2  
  
x1) -1  
x2) 2  
The system of equations is solved in 539 microseconds.

**Таблица зависимости времени работы от количества потоков:**

|  |  |
| --- | --- |
| Кол-во потоков | Время работы |
| 1 | 1630 микросекунд |
| 2 | 1402 микросекунды |
| 5 | 1151 микросекунда |
| 6 | 1081 микросекунда |
| 8 | 901 микросекунда |
| 10 | 868 микросекунд |
| 12 | 1104 микросекунды |

**Вывод**

В процессе работы я получил навыки работы с многопоточностью в ОС. Основными ошибками при выполнении лабораторной работы стали неверное вычисление множителя при произведении вычитания строк и неточности при выборе индексов строк для вычисления. Из-за первой проблемы после завершения работы программы исходный массив никак не изменялся, эта проблема была решена переработкой формул для вычисления индексов элемента. Вторая проблема была решена аналогичным образом. После проведения замеров времени я пришел к выводу, что основной прирост в производительности происходит при использовании 8 потоков. Дальнейшее увеличение числа потоков не дает серьезного прироста в производительности, а после 10 потоков, производительность начинает падать.