Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ

к лабораторной работе №1

на тему

**УПРАВЛЕНИЕ ПРОЦЕССАМИ, ПОТОКАМИ, НИТЯМИ**

Выполнил студент гр.253503 Царук В.А.

Проверил ассистент кафедры информатики Гриценко Н.Ю.

Минск 2024

**СОДЕРЖАНИЕ**

[**1 ФОРМУЛИРОВКА ЗАДАЧИ** 3](#_Toc178885047)

[**1.1 Цель работы** 3](#_Toc178885048)

[**2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ** 4](#_Toc178885049)

[**2.1 Процесс, поток, нити** 4](#_Toc178885050)

[**3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ** 5](#_Toc178885051)

[**3.1 Вывод программы** 5](#_Toc178885052)

[**ЗАКЛЮЧЕНИЕ** 6](#_Toc178885053)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 7](#_Toc178885054)

[**ПРИЛОЖЕНИЕ А (обязательное) Листинг программного кода** 8](#_Toc178885055)

# **1 ФОРМУЛИРОВКА ЗАДАЧИ**

### **1.1 Цель работы**

Целью выполнения данной лабораторной работы является возобновление, закрепление и развитие навыков программирования приложений Windows. Концепции вычислительных процессов, потоков, нитей; реализация в Windows. Основные этапы жизненного цикла процессов (потоков) и элементарное управление ими: порождение, завершение, получение и изменение состояния. Типичное (простое) использование многозадачности и многопоточности.

# **2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

### **2.1 Процесс, поток, нити**

Вычислительный процесс – это программа, выполняемая на процессоре. В операционной системе Windows каждый процесс рассматривается как системный объект, который представляет собой выполнение программы с набором ресурсов, включая память, дескрипторы и процессорное время. Состояния процесса включают: запуск (start), выполнение (running), ожидание (waiting) и завершение (terminated) [1].

Вычислительный поток – это наименьшая единица выполнения кода внутри процесса. В Windows каждый поток является системным объектом, создающимся операционной системой и управляемым планировщиком. Потоки разделяют адресное пространство и пространство дескрипторов процесса, к которому принадлежат, что позволяет им совместно работать над данными. Существует несколько состояний потоков: выполнение, готовность, ожидание, завершение и приостановка. Поток может быть приостановлен и возобновлен в любой момент [2].

Нити – это выполнения кода, похожие на потоки, но без прямой поддержки со стороны операционной системы. Их переключение происходит вручную в контексте одного потока. Нити работают в рамках одного потока и разделяют его ресурсы и переключение между ними осуществляется программно, что дает больше контроля над их выполнением [3].

# **3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ**

### **3.1 Вывод программы**

В результате выполнения лабораторной работы была написана программа которая создает три потока с разными приоритетами, в которых выполняется перемножение матриц. На рисунке 1.1 показан результат работы программы, где выводится время работы каждого потока.

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рисунок 1.1 – Консольный вывод работы программы

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения лабораторной работы были закреплены и развиты навыки программирования приложений Windows. Концепции вычислительных процессов, потоков, нитей; реализация в Windows. Основные этапы жизненного цикла процессов (потоков) и элементарное управление ими: порождение, завершение, получение и изменение состояния. Типичное (простое) использование многозадачности и многопоточности.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Класс Win32\_Process – Win32 apps | Microsoft Learn [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/cimwin32prov/win32-process

[2] A complete introduction to Windows OS processes, threads and related resources [Электронный ресурс]. – Режим доступа: https://www.tenouk.com/ModuleT.html

[3] Волокон – Win32 apps | Microsoft Learn [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/procthread/fibers

# **ПРИЛОЖЕНИЕ А (обязательное) Листинг программного кода**

Листинг 1 – Файл Source.cpp

#include <windows.h>

#include <iostream>

#include <vector>

#include <chrono>

#define THREAD\_COUNT 3

struct ThreadData {

int threadID;

int priority;

unsigned long long iterations;

double executionTime;

};

void MatrixMultiplication(int size) {

std::vector<std::vector<int>> matrixA(size, std::vector<int>(size, 15));

std::vector<std::vector<int>> matrixB(size, std::vector<int>(size, 20));

std::vector<std::vector<int>> result(size, std::vector<int>(size, 25));

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

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

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

result[i][j] += matrixA[i][k] \* matrixB[k][j];

}

}

}

}

DWORD WINAPI ThreadFunction(LPVOID lpParam) {

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

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

for (unsigned long long i = 0; i < data->iterations; ++i) {

MatrixMultiplication(100); // Умножение матриц размером 100x100

if (i % 10 == 0) {

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

std::chrono::duration<double> elapsed = current - start;

std::cout << "Thread " << data->threadID << " is running. Time elapsed: "

<< elapsed.count() << " seconds." << std::endl;

}

}

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

std::chrono::duration<double> elapsed = end - start;

data->executionTime = elapsed.count();

return 0;

}

int main() {

HANDLE threads[THREAD\_COUNT];

ThreadData threadData[THREAD\_COUNT];

int priorities[THREAD\_COUNT] = { THREAD\_PRIORITY\_LOWEST, THREAD\_PRIORITY\_NORMAL, THREAD\_PRIORITY\_HIGHEST };

for (int i = 0; i < THREAD\_COUNT; ++i) {

threadData[i].threadID = i + 1;

threadData[i].priority = priorities[i];

threadData[i].iterations = 100;

threadData[i].executionTime = 0.0;

threads[i] = CreateThread(

nullptr,

0,

ThreadFunction,

&threadData[i],

0,

nullptr

);

if (threads[i] == nullptr) {

std::cerr << "Error in creating thread " << i + 1 << std::endl;

return 1;

}

SetThreadPriority(threads[i], priorities[i]);

}

WaitForMultipleObjects(THREAD\_COUNT, threads, TRUE, INFINITE);

for (int i = 0; i < THREAD\_COUNT; ++i) {

std::cout << "Thread " << threadData[i].threadID

<< " with priority " << threadData[i].priority

<< " ended. Time of work: "

<< threadData[i].executionTime << " seconds." << std::endl;

CloseHandle(threads[i]);

}

return 0;

}