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.253504 Сапроненко В.В.

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2024

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc177388377)

[2 Краткие теоритические сведения 4](#_Toc177388378)

[3 Описание функций программы 5](#_Toc177388379)

[3.1 Результат выполнения программы. 5](#_Toc177388381)

Заключение 6

[Список использованных источников 7](#_Toc177388383)

[Приложение А (обязательное) Исходный код программы 8](#_Toc177388384)

# **1 ФОРМУЛИРОВКА ЗАДАЧИ**

Целью выполнения лабораторной работы возобновление, закрепление и развитие навыков программирования приложений Windows. В ходе выполнения работы необходимо изучить концепции вычислительных процессов, потоков, нитей и их реализацию в Windows; основные этапы жизненного цикла процессов (потоков) и элементарное управление ими: порождение, завершение, получение и изменение состояния; Типичное (простое) использование многозадачности и многопоточности.

В качестве задачи необходимо запустить несколько потоков с разными приоритетами и оценить их производительность:

– запуск нескольких потоков с заранее заданными приоритетами;

– оценка производительности этих потоков;

– отображение итогового времени работы потока после завершения;

– отображение объёма выполненной работы (счётчик итераций);

2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Приложение состоит из одного или нескольких процессов. Процесс, в простейших терминах, — это выполняющаяся программа. Один или несколько потоков выполняются в контексте процесса. Поток — это базовая единица, которой операционная система выделяет процессорное время. Поток может выполнять любую часть кода процесса, включая те части, которые в настоящее время выполняются другим потоком.

Объект задания позволяет управлять группами процессов как единое целое. Объекты заданий — это именуемые, защищаемые, общие объекты, управляющие атрибутами связанных с ними процессов. Операции, выполняемые с объектом задания, влияют на все процессы, связанные с объектом задания.

Пул потоков — это коллекция рабочих потоков, которые эффективно выполняют асинхронные обратные вызовы от имени приложения. Пул потоков в основном используется для уменьшения количества потоков приложения и обеспечения управления рабочими потоками.

Нить — это единица выполнения, которую приложение должно запланировать вручную. Нити выполняются в контексте потоков, которые планируют их. [1]

3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

Согласно формулировке задачи, были спроектированы следующие функции программы:

– запуск нескольких потоков с разными приоритетами; [2]

– оценка производительности потоков, включая отображение итогового времени после завершения, а также объема выполненной потоком работы с помощью счетчика итераций; [3]

– выполнение сортировки массива со случайными значениями в каждом потоке;

## **Результат выполнения программы**

Вывод результатов работы потоков с различными приоритетами, включая отображение итоговых результатов по завершению каждого потока (рисунок 3.1).

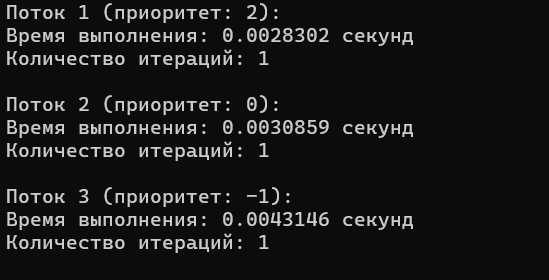
****

Рисунок 3.1 – Результат выполнения программы

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения задания были рассмотрены основные аспекты управления потоками и процессами в операционной системе Windows с использованием Windows API. Были созданы несколько потоков с разными приоритетами, каждому из которых была поставлена задача, требующая вычислительных ресурсов — сортировка большого массива данных. В процессе работы была изучена и реализована установка приоритетов для потоков, их запуск, контроль выполнения, а также получение итоговых результатов работы, таких как время выполнения и количество итераций.

Данное задание позволило не только возобновить и закрепить навыки работы с многопоточностью, но и проанализировать, как приоритет потоков влияет на их производительность. Было установлено, что высокоприоритетные потоки завершаются быстрее, что подтверждает влияние планировщика Windows на распределение ресурсов между потоками. Реализованное решение наглядно демонстрирует основные этапы жизненного цикла потоков — их создание, выполнение и завершение.

Полученные знания и практические навыки позволяют глубже понимать принципы многопоточности и многозадачности в Windows, а также применить эти концепции в будущих проектах для оптимизации работы приложений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Build desktop Windows apps using the Win32 API [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/

[2] Основные сообщения ОС Windows (Win32 API). Программирование в ОС Windows. Лекция 1 [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=wTArIolxch0

[3] Разработка приложений с помощью WinAPI [Электронный ресурс]. – Режим доступа: https://shorturl.at/BDJW8

ПРИЛОЖЕНИЕ А

(обязательное)

**Исходный код программы**

#include <iostream>

#include <windows.h>

#include <vector>

#include <algorithm>

#include <ctime>

struct ThreadData {

int priority;

LARGE\_INTEGER startTime, endTime;

int iterations;

};

DWORD WINAPI ThreadFunction(LPVOID lpParam) {

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

QueryPerformanceCounter(&data->startTime);

const int size = 10000;

std::vector<int> arr(size);

for (int& val : arr) {

val = rand() % 10000;

}

std::sort(arr.begin(), arr.end());

data->iterations = 1;

QueryPerformanceCounter(&data->endTime);

return 0;

}

void SetThreadPriorityAndStart(HANDLE& hThread, int priority, ThreadData& data) {

hThread = CreateThread(

NULL,

0,

ThreadFunction,

&data,

0,

NULL

);

if (hThread == NULL) {

std::cerr << "Ошибка создания потока!" << std::endl;

exit(1);

}

if (!SetThreadPriority(hThread, priority)) {

std::cerr << "Ошибка установки приоритета!" << std::endl;

}

}

int main() {

setlocale(LC\_ALL, "ru");

srand(static\_cast<unsigned>(time(0)));

const int numThreads = 3;

ThreadData data[numThreads];

HANDLE hThreads[numThreads];

int priorities[numThreads] = { THREAD\_PRIORITY\_HIGHEST, THREAD\_PRIORITY\_NORMAL, THREAD\_PRIORITY\_BELOW\_NORMAL };

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

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

SetThreadPriorityAndStart(hThreads[i], priorities[i], data[i]);

}

WaitForMultipleObjects(numThreads, hThreads, TRUE, INFINITE);

LARGE\_INTEGER frequency;

QueryPerformanceFrequency(&frequency);

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

double duration = static\_cast<double>(data[i].endTime.QuadPart - data[i].startTime.QuadPart) / frequency.QuadPart;

std::cout << "Поток " << i + 1 << " (приоритет: " << data[i].priority << "):" << std::endl;

std::cout << "Время выполнения: " << duration << " секунд" << std::endl;

std::cout << "Количество итераций: " << data[i].iterations << std::endl << std::endl;

CloseHandle(hThreads[i]);

}

return 0;

}