# **Лабораторная работа 1: Основы потоков и процессов в Windows**

Цели работы:

1. Изучить основы работы с потоками и процессами в Windows.
2. Научиться создавать и управлять потоками.
3. Реализовать простую программу, использующую процессы.

Задание:

Создайте программу, которая запускает блокнот (notepad.exe) в отдельном процессе и создает поток, выводящий сообщение в консоль. Используйте функции Windows API для создания процесса и потока.

Код:

#include <windows.h>

#include <iostream>

using namespace std;

// Функция для создания нового процесса

void CreateNewProcess(const wchar\_t\* applicationName) {

STARTUPINFO si; // Структура для хранения информации о процессе

PROCESS\_INFORMATION pi; // Структура для хранения информации о созданном процессе

ZeroMemory(&si, sizeof(si)); // Обнуляем структуру STARTUPINFO

si.cb = sizeof(si); // Указываем размер структуры

ZeroMemory(&pi, sizeof(pi)); // Обнуляем структуру PROCESS\_INFORMATION

// Используем CreateProcessW для создания процесса с широкими строками

if (!CreateProcess(applicationName, NULL, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi)) {

wcout << L"Ошибка создания процесса." << std::endl; // Выводим сообщение об ошибке

return; // Выходим из функции, если создание процесса не удалось

}

// Ожидаем завершения процесса

WaitForSingleObject(pi.hProcess, INFINITE);

wcout << L"Процесс завершен." << std::endl; // Сообщаем о завершении процесса

// Закрываем дескрипторы процесса и потока

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

}

// Функция, которая будет выполняться в потоке

DWORD WINAPI ThreadFunction(LPVOID lpParam) {

cout << "Поток выполняется." << std::endl; // Выводим сообщение о выполнении потока

return 0; // Возвращаем 0, чтобы указать на успешное завершение

}

// Функция для создания и выполнения потока

void CreateThreadExample() {

HANDLE hThread; // Дескриптор потока

hThread = CreateThread(NULL, 0, ThreadFunction, NULL, 0, NULL); // Создаем новый поток

// Ожидаем завершения потока

WaitForSingleObject(hThread, INFINITE);

CloseHandle(hThread); // Закрываем дескриптор потока

}

int main() {

// Полный путь к блокноту

setlocale(0, "");

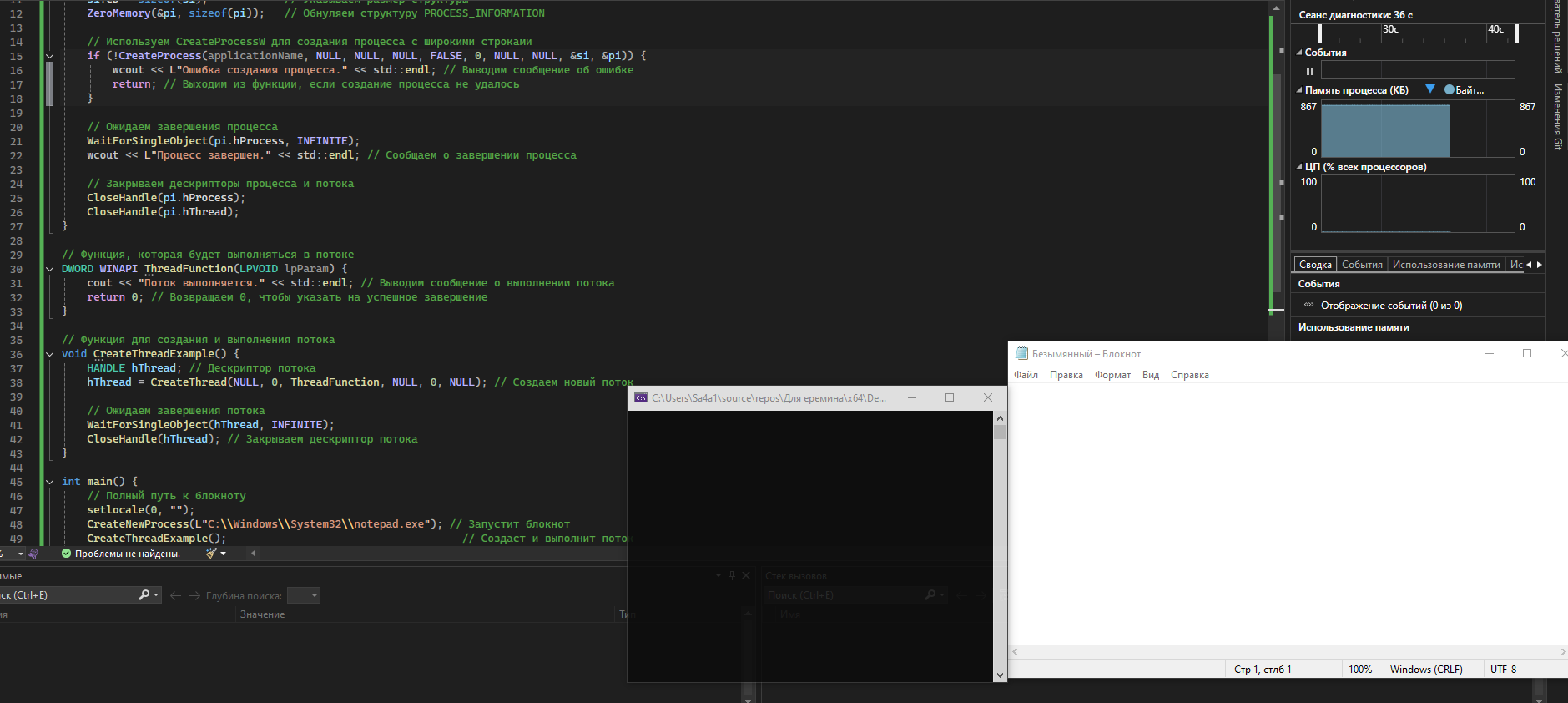
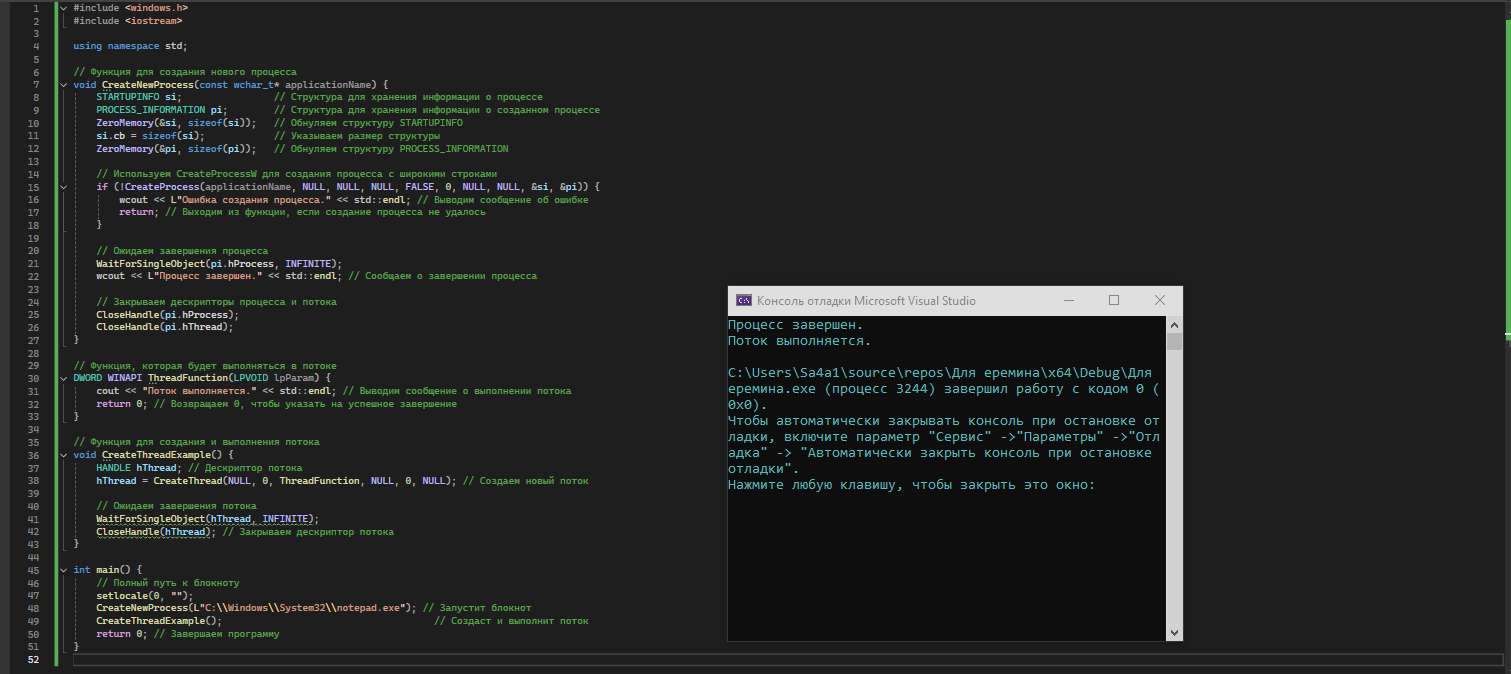
CreateNewProcess(L"C:\\Windows\\System32\\notepad.exe"); // Запустит блокнот

CreateThreadExample(); // Создаст и выполнит поток

return 0; // Завершаем программу

}

Результат:



# **Контрольные вопросы**

1. Что такое поток и процесс в контексте операционной системы?

Поток — это последовательность инструкций программы, которые выполняются процессором параллельно с другими потоками.

Процесс — исполняемое на компьютере приложение вместе со всеми его ресурсами, необходимыми для исполнения.

1. Как создать новый процесс в Windows с использованием WinAPI

Новый процесс с использованием WinAPI можно создать, используя функцию CreateProcess(). Она принимает несколько параметров, таких как имя выполняемого файла, параметры командной строки, атрибуты процесса и потока, а также флаги создания.

1. В чем разница между функциями CreateThread и CreateProcess?

CreateProcess – создает новый отдельный процесс с собственными ресурсами, а CreateThread – создает поток внутри существующего процесса.

1. Как использовать функцию WaitForSingleObject в многопоточном программировании?

Функция WaitForSingleObject проверяет текущее состояние указанного объекта. Если объект находится в состоянии без знака, вызывающий поток переходит в состояние ожидания до тех пор, пока объект не будет помечен или не истекает интервал времени ожидания (пример использования функции в коде).

1. Почему важно закрывать дескрипторы потоков и процессов?

Дескрипторы необходимо закрывать во избежание утечек памяти и для оптимизации использования системных ресурсов. Так же большое количество открытых дескрипторов может привести к нестабильности системы и снижению ее производительность.

1. Как передать параметры в поток?

Передача параметров в поток через функцию CreateThread в Windows осуществляется с использованием параметра lpParameter. Этот параметр позволяет передать указатель на данные, которые необходимо использовать в функции потока.

1. Что произойдет, если не дождаться завершения процесса перед закрытием приложения?

Если не дождаться завершения процессов или потоков перед закрытием приложения, это может привести к утечке системных ресурсов, неполным операциям, потере данных и непредсказуемому поведению приложения. Потоки или процессы могут не успеть корректно освободить ресурсы или завершить важные задачи, такие как запись данных, что особенно критично. Также возможны проблемы с синхронизацией, такие как взаимоблокировки. Поэтому важно обеспечить корректное завершение всех потоков и процессов перед закрытием приложения, используя механизмы ожидания, такие как WaitForSingleObject или WaitForMultipleObjects в Windows.