БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий

и робототехники

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

О Т Ч Ё Т

о лабораторной работе № 6

Дисциплина

«Системное программирование»

Тема

«Управление потоками в Windows»

Выполнил: студент гр. 10702217 Храмков Д. С.

Проверил: Разорёнов Н. А.

Минск 2019

***Лабораторная работа № 6***

**УПРАВЛЕНИЕ ПОТОКАМИ В WINDOWS**

**Цель работы:** изучить основы создания и управления потоками в ОС Windows.

**Изучаемые вопросы**

1. Виды потоков, состояния потока.
2. Структура CONTEXT.
3. Создание потока.
4. Относительный приоритет потока.
5. Потоковая функция.
6. Функции WinAPI для управления потоками.
7. Окончание потока.
8. Время выполнения потока.

***Постановка задачи***

Разработать многопоточное Win32-приложение, которое использует диалоговое окно для управления потоками процессов. Дизайн диалогового окна задается вариантом, преподавателем или самостоятельно. Для визуализации работы потоков использовать соответствующие элементы управления диалога, графику. Приложение должно содержать три потока. Предусмотреть вывод системной информации о потоках (например, полей CONTEXT, временные параметры и т.д.). В отчете привести диаграмму состояния потоков, копии окон.

1. **Виды потоков, состояния потока.**

**Виды потоков делятся на:**

* **Потоки на уровне пользователя;**
* **Потоки на уровне ядра;**
* **Комбинированная модель.**

**У потоков существуют следующие состояния:**

* **Рабочее;**
* **Нерабочее (простаивающее).**

if (hThread[0] != NULL) {

SuspendThread(hThread[0]);

}

if (hThread[0] != NULL) {

ResumeThread(hThread[0]);

}

1. **Структура CONTEXT.**

**Структура CONTEXT содержит данные о состоянии регистров с учётом специфики конкретного процессора. CONTEXT используется системой для выполнения различных внутренних операций.**

CONTEXT context;

GetThreadContext(hThread[0], &context);

1. **Создание потока.**

**Поток создаётся при помощи функции CreateThread().**

hThread[0] = CreateThread(NULL, 0, threadFunc1, &threadParam, 0, &threadID[0]);

1. **Относительный приоритет потока.**

**Существуют следующие приоритеты потоков:**

* THREAD\_PRIORITY\_BELOW\_NORMAL;
* THREAD\_PRIORITY\_NORMAL;
* THREAD\_PRIORITY\_ABOVE\_NORMAL;
* THREAD\_PRIORITY\_HIGHEST;
* THREAD\_PRIORITY\_TIME\_CRITICAL;
* THREAD\_PRIORITY\_LOWEST;
* THREAD\_PRIORITY\_IDLE.

Чтобы присвоить один из вышеперечисленных приоритетов потоку, надо задать его в функции SetPriorityThread().

if (hThread[0] != NULL) {

SetThreadPriority(hThread[0], THREAD\_PRIORITY\_ABOVE\_NORMAL);

}

1. **Потоковая функция.**

**Потоковой называется функция, запускающаяся при создании потока. Для запуска функции необходимо указать её имя при создании потока.**

hThread[2] = CreateThread(NULL, 0, threadFunc3, &threadParam, 0, &threadID[2]);

1. **Функции WinAPI для управления потоками.**

**Для управления потоками в WinAPI существуют следующие команды:**

* **CreateThread();**
* **SetThreaadPriority();**
* **ResumeThread();**
* **SuspendThread();**
* **GetThreadTimes();**
* **WaitForMultipleObjects();**
* **GetThreaadContext();**
* **ExitThread();**
* **TerminateThread();**

GetThreadTimes(hThread[i], &lpCreationTime, &lpExitTime, &lpKernelTime, &lpUserTime);

ExitThread(0);

1. **Окончание потока.**

**Существует несколько способов завершить поток:**

* **Функция потока возвращает управление;**
* **Поток самоуничтожается вызовом функции ExitThread;**
* **Один из потоков данного или стороннего процесса вызывает функцию TerminateThread;**
* **Завершается процесс, содержащий данный поток;**
* **Какой-либо поток процесса вызывает функцию ExitProcess();**
* **Какой-либо поток вызывает функцию TerminateThread() с дескриптором потока.**
* **Какой-либо поток вызывает функцию TerminateProcess() с дескриптором процесса.**

**Самым «здоровым» способом является первый из вышеперечисленных, тем не менее применяются и другие в зависимости от ситуации.**

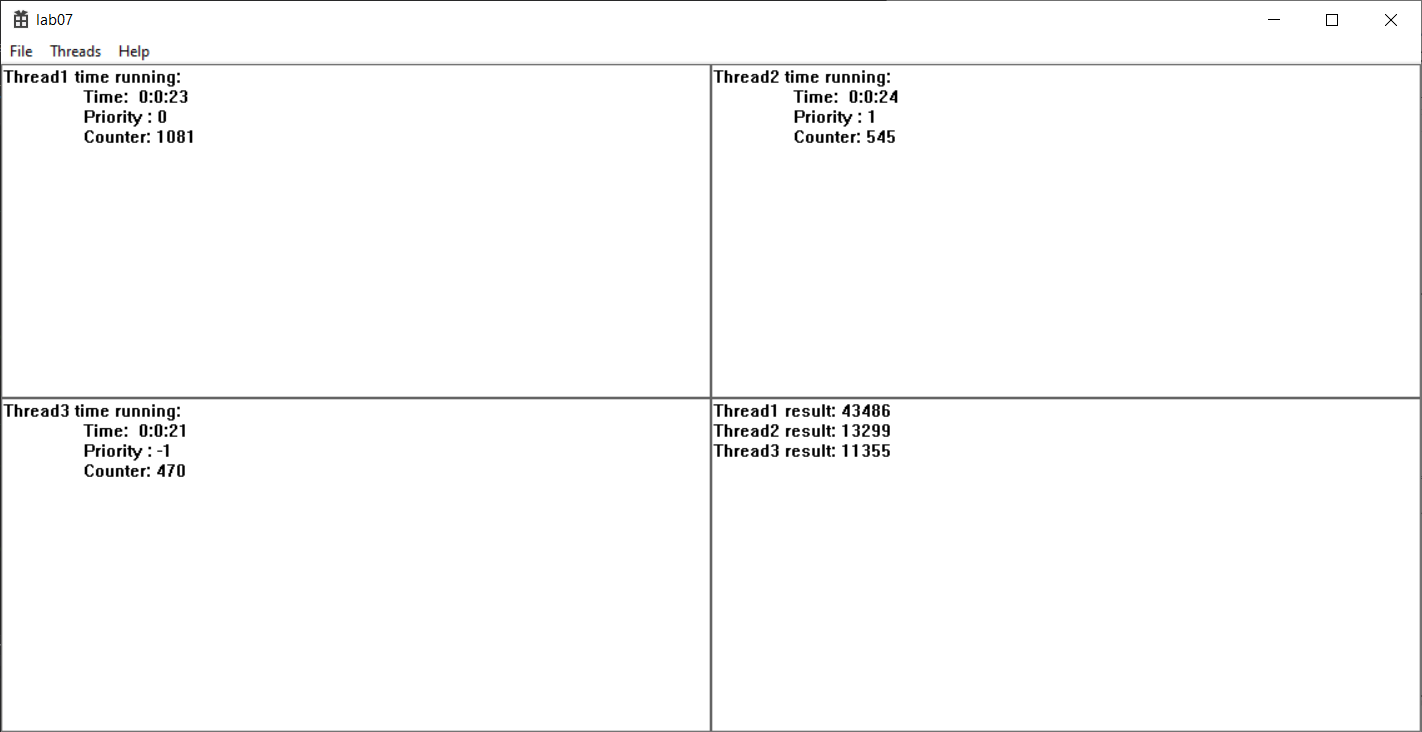
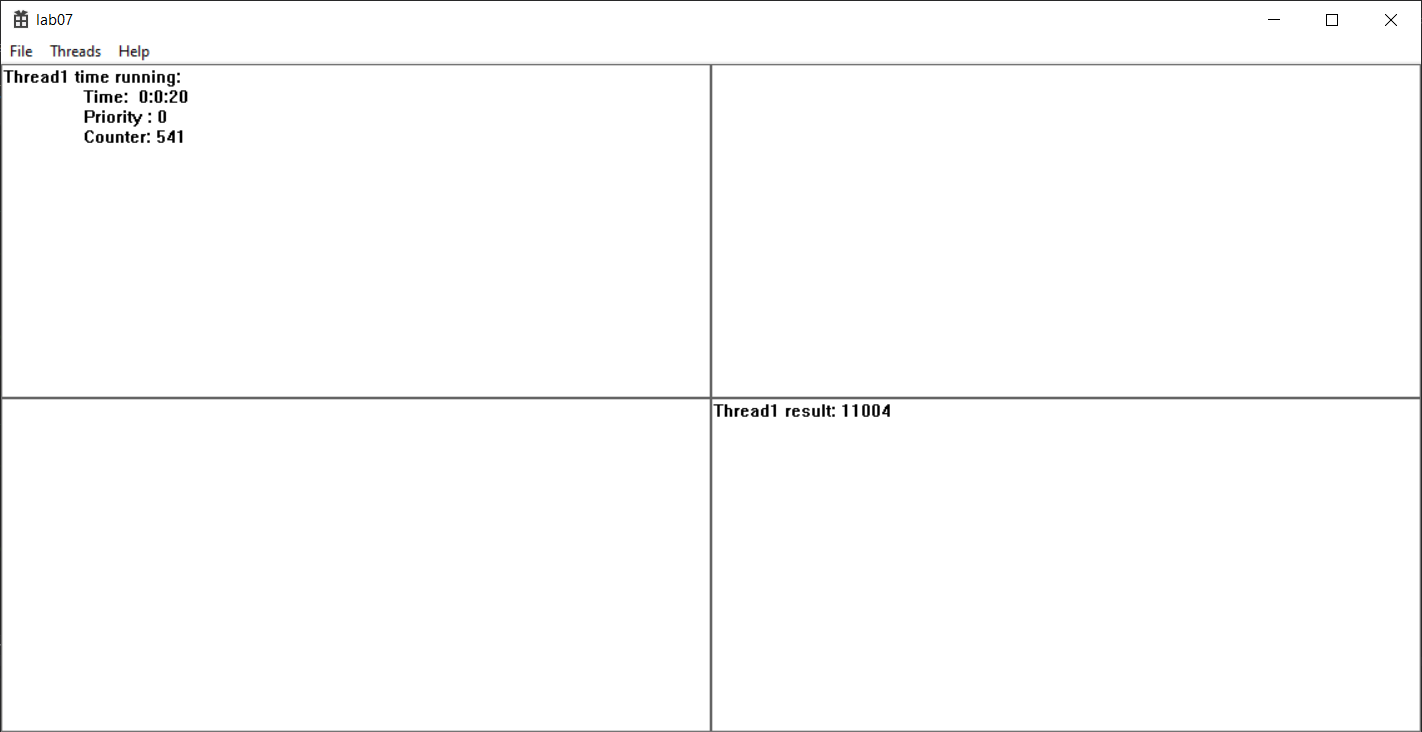
ExitThread(0);

1. **Время выполнения потока.**

**Чтобы получить время выполнения потока используется функция GetThreadTimes, которая возвращает время создания, завершения, выполнения и время ядра.**

GetThreadTimes(hThread[i], &creationTime, &exitTime, &kernelTime, &userTime);

**Результаты работы программы:**



**Выводы:**

1. **Научились создавать потоки;**
2. **Научились получать потоковое время;**
3. **Научились задавать приоритет потоков;**
4. **Научились получать информацию о потоке.**