Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Поволжский государственный технологический университет»

Факультет информатики и вычислительной техники

Кафедра ИБ

Отчет по лабораторной работе №5

по дисциплине «Безопасность операционных систем»

Выполнил: обучающийся гр. БИ-31  
Шестаков А.Д.

(подпись) (Ф.И.О.)

Проверил: ст. преподаватель  
Кречетов А.А.

(подпись) (Ф.И.О.)

Йошкар-Ола  
2022 г.

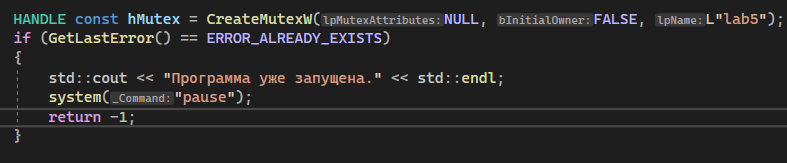
**Цель работы:** разработать консольное приложение, создающее потоки управления. Каждый поток управления периодически выводит на консоль информацию.

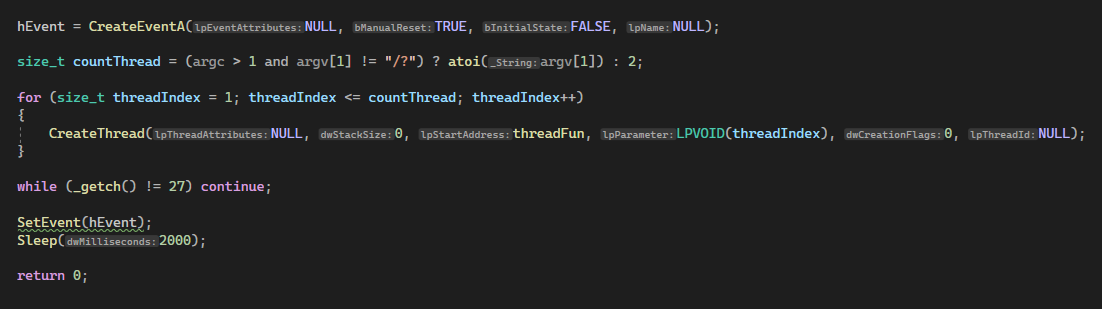
Информация об используемых функциях:

1. **CreateThread -** создаёт поток, работающий в виртуальном адресном пространстве вызывающего процесса.
2. **CreateEvent –** создаёт или открывает именованный или безымянный объект события.
3. **InitializeCriticalSection –** инициализирует объект критической секции.
4. **CreateMutex –** создаёт или открывает именнованный или безымянный объект мьютекса.
5. **WaitForSingleObject –** ожидает, пока событие объекта не перейдёт в сигнальное состояние или пока не истечёт время ожидания.

При запуске данная программа выводит информацию с помощью функции **Info()**, если соответственный параметр указан.



Создаётся взаимоисключение (**mutex**) которое не даст создать ещё один процесс данной программы:

Для того, чтобы поток узнал о том, что пора завершить работу, создаётся событие (**Event**), который после срабатывания (**SetEvent**) будет сигнализировать функции **WaitForSingleObject** что пора завершить поток. Так как событие едино для всех потоков, то при срабатывании его, будут завершены все потоки, и программа завершит свою работу:

В качестве того, что должен делать поток, написана следующая функция:

Для синхронизации потоков используется отдельная критическая секция, она была инициализирована в **main(),** а в данном случае поток резервирует для своей работы данную секцию. Поток уведомляет, что он вызвался и выходит из критической секции, освобождая место для другого потока.

**Вывод:** изучил основные механизмы синхронизации параллельных ветвей выполнения (потоков) в операционных системах с использованием базовых примитивов OC.