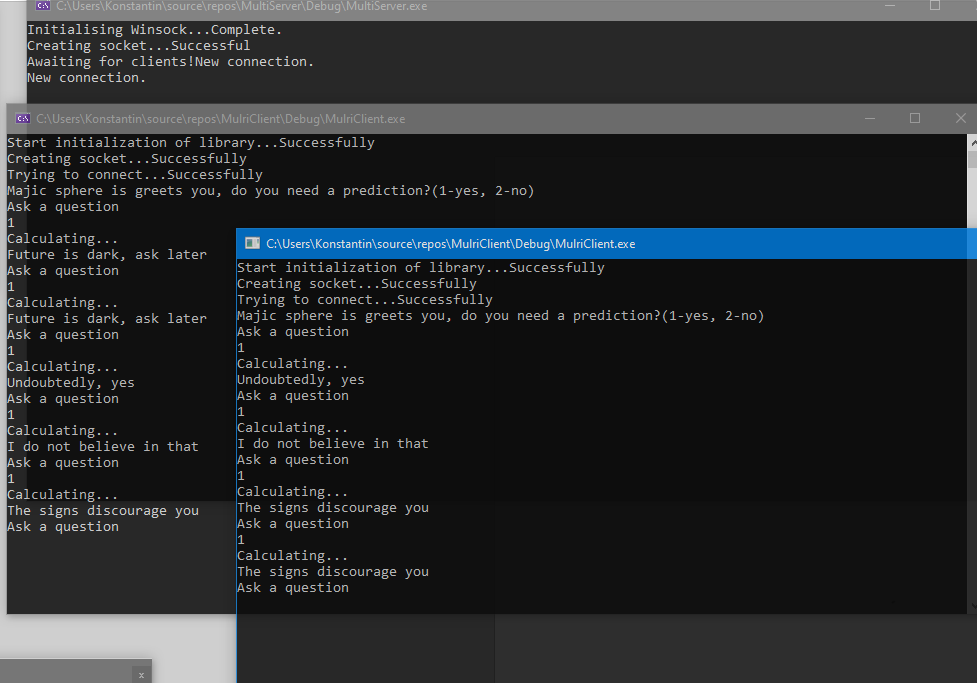
Пример работы многопоточного сервера и нескольких клиентов:

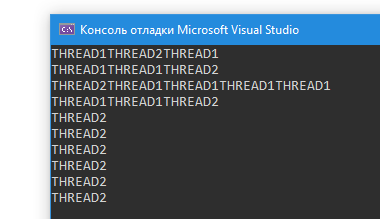


Для каждого нового клиента создаётся новый поток, который продолжает и продолжает с ним общаться. Код находится в соответствующих папках.

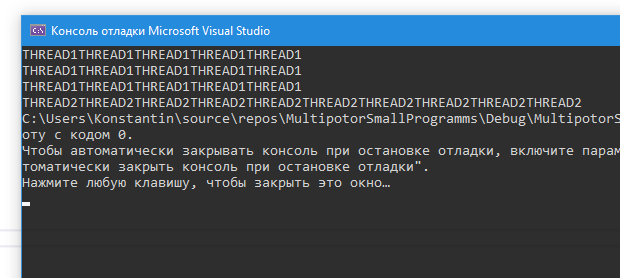
Работа с многопоточностью

Несинхронизированные потоки.

В целом, описывать особо нечего: основной поток в main выписывает THREAD1, а побочный поток выписывает TRHEAD2. Если бы потоки были синхронизированы, то THREAD1 и THREAD2 выписывались бы одновременно.

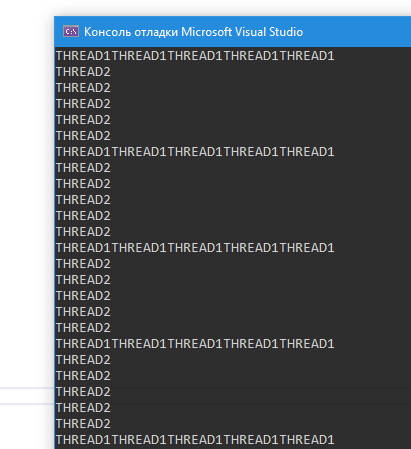


Критическая секция



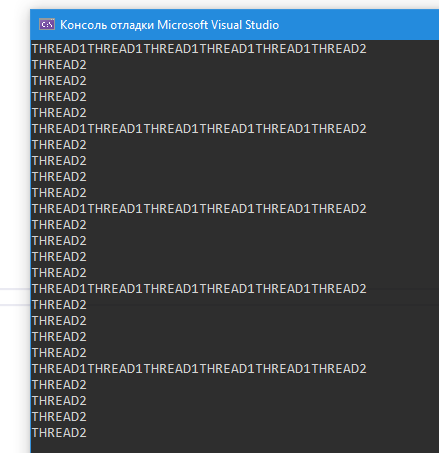
Код сначала занимает критическую секцию для потока main, затем, после того как она проходит цикл, стоящие потоки, ожидавшие доступ к критической секции, получают их. Поэтому сначала выписывается три строчки THREAD1, по количеству обходов цикла, а затем выписывается два потока THREAD2(два, потому что большее количество потоков не создаётся).

Мьютекс



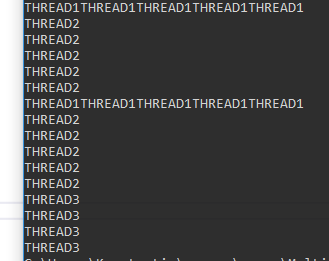
Имея один тот же алгоритм, в разных реализациях получается разный ответ. Мьютексы не передают «право» на пользование какой-то областью памяти, как критические секции, а просто ожидают тех, кто к ним обратится. Создав мьютекс, передав его в другой поток, они с основным дальше только менялись местами: как только один поток освобождал мьютекс, другой тут же помечал его как занятый и наоборот.

События



Мы изначально поставили два события, один на один поток, другой – на другой. Как только в основном потоке срабатывало событие, оно выписывало THREAD1 и переводило hEvent2 в сигнальное состояние. Второй поток это видел, выписывал THREAD2 и переводил hEvent1 в сигнальное состояние, что видел основной поток. Они обменивались сигналами пока шил циклы.   
Если бы было несколько потоков, которые слушали те же hEvent1 и hEvent2, то они бы тоже реагировали каким-то заложенным в них кодом на приведение события в сигнальное состояние.

Семафоры



В одном и том же потоке, поставив одну из «нитей» на семафор, мы продолжали подключать остальные. Но лимит семафора не позволял подключиться другим функциям, поэтому он позволял одновременно работать только одному потоку, которые выписывали THREAD1 и THREAD2, но под конец, когда оба они освободили семафор, третий поток, THREAD3, смог подключиться, но проработал только один раз. В этом отличие семафоров от остальных типов синхронизации, он даёт возможность использовать себя только какому-то определённому количеству потоков