# План обучения (Неделя 3): Продвинутая многопоточность и асинхронность

Общая нагрузка: 30 часов (6 часов/день).

Цель: Переход от мышления "Game Loop" к конкурентной обработке запросов и асинхронной архитектуре.

## День 1: Физика процессов и ThreadPool

**Фокус:** Понимание стоимости абстракций. Почему мы перестаем создавать new Thread() и переходим к управлению задачами.

### Теория (3 часа) [Теория C# Неделя 3 День 1: Потоки и ThreadPool](https://docs.google.com/document/d/1KrPj5AByAphaMyBA5tMUSc_dGnDSx1W9V-bsBeFC3A4/edit?usp=sharing)

1. **Анатомия потока:**
   * Различия между потоком ОС и управляемым потоком (Managed Thread).
   * Аллокация стека (1MB) и переключение контекста (Context Switch). Почему 1000 потоков "убивают" процессор.1
2. **ThreadPool CLR:**
   * Устройство пула: Глобальная очередь vs Локальные очереди.
   * Алгоритм Work-Stealing и Hill Climbing: как.NET балансирует нагрузку.3
   * Различие между IO-Bound и CPU-Bound операциями: когда использовать Task.Run, а когда нет.1
3. **Task Parallel Library (TPL):**
   * Класс Task как "обещание" (Future), а не поток.
   * Статусы задач: Created, Running, RanToCompletion, Faulted, Canceled.2

### Практика (3 часа)

* **Лабораторная работа "Cost of Concurrency":**
  + Написать бенчмарк (BenchmarkDotNet), сравнивающий создание 10,000 потоков (new Thread) и 10,000 задач (Task.Run).
  + Зафиксировать потребление памяти и время выполнения.
* **Анализ пула:**
  + Использование ThreadPool.GetAvailableThreads и ThreadPool.SetMinThreads для наблюдения за поведением пула под нагрузкой.

## День 2: Внутреннее устройство Async/Await

**Фокус:** Демистификация "магии" компилятора. Как работает код, когда поток освобождается.

### Теория (3 часа) [Async/Await: Теория и Практика C# день 2](https://docs.google.com/document/d/1y-tje53uyOhVTCe9tthXMAS4VpPThLfYx3uNHU_e0eA/edit?usp=sharing)

1. **State Machine (Машина состояний):**
   * Как компилятор преобразует метод async в структуру IAsyncStateMachine.
   * "Поднятие" (Hoisting) локальных переменных в поля структуры: влияние на GC и память.4
2. **Обработка исключений в асинхронности:**
   * Разница между async Task и async void. Почему async void — это "Crash the Process".4
   * Агрегирование исключений: AggregateException при использовании Task.WhenAll.
3. **Контекст синхронизации (SynchronizationContext):**
   * Почему в ASP.NET Core его нет, а в Legacy (WPF/Unity) он есть.
   * Опасность ConfigureAwait(true) в библиотеках и риск дедлоков (Deadlocks).1

### Практика (3 часа)

* **Decompilation Review:**
  + Написать простой async-метод и разобрать его через dotPeek/SharpLab, найдя метод MoveNext и поля состояния.
* **Refactoring:**
  + Взять "лапшеобразный" код на колбэках (стиль Unity Coroutines или EAP) и переписать на чистый async/await.
  + Реализовать правильную обработку ошибок через try-catch внутри async методов.

## День 3: Безопасность данных и примитивы синхронизации

**Фокус:** Защита разделяемого состояния. Как избежать Race Conditions, не убив производительность.

### Теория (3 часа) [Теория синхронизации и безопасность данных день 3](https://docs.google.com/document/d/13Cru-oWeQMMwebLlRYIjrhAhn7yhlkPuoguvEdeQZ2w/edit?usp=sharing)

1. **Проблематика гонок (Race Conditions):**
   * Атомарность операций. Почему i++ не атомарен.
   * Модели памяти и переупорядочивание инструкций процессором (Memory Barriers, volatile).5
2. **Блокировки (Locking):**
   * Interlocked: самые быстрые операции (CAS - Compare And Swap).
   * Monitor (lock): гибридная блокировка.
   * SpinLock: когда ожидание дешевле переключения контекста.5
3. **Запрет на блокировки в Async:**
   * Почему нельзя использовать lock внутри await. Понятие Thread Affinity.6

### Практика (3 часа)

* **Симуляция банковского счета:**
  + Создать класс счета с ошибкой Race Condition.
  + Воспроизвести ошибку многопоточным тестом.
  + Исправить с использованием Interlocked (для простых типов) и lock (для логики).
* **Deadlock Simulation:**
  + Искусственно создать взаимную блокировку двух потоков и проанализировать дамп потоков в отладчике VS.

## День 4: Асинхронные паттерны и Троттлинг

**Фокус:** Управление конкурентностью. Как обрабатывать тысячи запросов, не перегружая систему.

### Теория (3 часа) [Теория Четвертого Дня Обучения](https://docs.google.com/document/d/1dCI_IsW8Ug04Mkq_iQAZVuR8zs5dOF_GWYWHZL4nft0/edit?usp=sharing)

1. **Продвинутая синхронизация:**
   * SemaphoreSlim: асинхронный семафор для ограничения доступа к ресурсу.
   * Паттерн "Producer-Consumer" с использованием Channel<T>.
2. **Отмена операций (Cancellation):**
   * Прокидывание CancellationToken.
   * Разница между "мягкой" и "жесткой" отменой (ThrowIfCancellationRequested vs проверка свойства).4
   * Работа с LinkedTokenSource для каскадной отмены.
3. **Паттерны параллелизма:**
   * Троттлинг (Throttling) и Bulkhead: защита внешних API от DDoS своими же запросами.

### Практика (3 часа)

* **Реализация "Smart Downloader" (Задание А):**
  + Написать сервис, загружающий список URL.
  + Ограничить параллелизм до N потоков с помощью SemaphoreSlim.
  + Реализовать полную поддержку отмены через CancellationToken.

## День 5: Сетевое взаимодействие (Low Level)

**Фокус:** Работа с "сырыми" данными. Разрушение мифа о том, что "один Send = один Receive".

### Теория (3 часа) [Теория сетевого взаимодействия C# день 5](https://docs.google.com/document/d/127UFo3mSv-8tMUTSGGHiWvof5ipU6g6cL00NKKU8Peo/edit?usp=sharing)

1. **Потоковая природа TCP:**
   * Понятие Stream. Почему TCP не гарантирует границы сообщений.
   * Фрагментация (Segmentation) и склейка пакетов (Coalescing).
2. **Работа с NetworkStream:**
   * Методы ReadAsync и WriteAsync.
   * Обработка частичного чтения (Partial Read): паттерн "Read Loop".7
3. **Протоколы прикладного уровня:**
   * Зачем нужны заголовки длины (Length-prefixing) или разделители (Delimiters) для парсинга сообщений.

### Практика (3 часа)

* **Создание TCP-клиента (Задание Б):**
  + Написать консольный Telnet-клиент.
  + Реализовать бесконечный цикл чтения в отдельном Task.
  + Обработать корректное закрытие сокета и сценарий разрыва соединения сервером (чтение 0 байт).

#### Источники

1. C# Threading: From Basic to Advanced | by Laks Tutor - Medium, дата последнего обращения: декабря 3, 2025, <https://medium.com/@lakstutor/c-threading-from-basic-to-advanced-84927e502a38>
2. Parallel Programming with SemaphoreSlim in .NET - C# Corner, дата последнего обращения: декабря 3, 2025, <https://www.c-sharpcorner.com/article/parallel-programming-with-semaphoreslim-in-net/>
3. Threading in C# - Part 4 - Advanced Threading - Joseph Albahari, дата последнего обращения: декабря 3, 2025, <https://www.albahari.com/threading/part4.aspx>
4. Concurrency in C# Cookbook, 2nd Edition - O'Reilly, дата последнего обращения: декабря 3, 2025, <https://www.oreilly.com/library/view/concurrency-in-c/9781492054498/>
5. Efficient Synchronization in C# with SemaphoreSlim - Oleg Kyrylchuk, дата последнего обращения: декабря 3, 2025, <https://okyrylchuk.dev/blog/efficient-synchronization-in-csharp-with-semaphoreslim/>
6. TCP client not reading the send half of the message from server - Stack Overflow, дата последнего обращения: декабря 3, 2025, <https://stackoverflow.com/questions/75206719/tcp-client-not-reading-the-send-half-of-the-message-from-server>
7. Part 4: Async & Parallel Programming – C# / .NET Interview Questions and Answers, дата последнего обращения: декабря 3, 2025, <https://bool.dev/blog/detail/c-net-interview-questions-and-answers-part-4-async-parallel-programming>
8. Receiving incomplete data when using SuperSimpleTcp for C# : r/dotnet - Reddit, дата последнего обращения: декабря 3, 2025, <https://www.reddit.com/r/dotnet/comments/15254bd/receiving_incomplete_data_when_using/>