**1. Что такое процесс, домен, поток? Как они связаны между собой?**

**Процесс** представляет собой экземпляр выполняющейся программы в операционной системе.

**Домен** приложения - это изолированная среда выполнения внутри процесса .NET, которая содержит и выполняет одно или несколько приложений.

**Поток** представляет собой наименьшую единицу исполнения внутри процесса. Он выполняет код в контексте процесса, совместно используя его ресурсы, но имея собственный указатель инструкций и стек вызовов.

**Связь :**

* Один процесс может содержать несколько доменов приложения.
* Внутри каждого домена приложения может выполняться несколько потоков, совместно использующих ресурсы этого домена.

**2. Как получить информацию о процессах?**

В .NET существует несколько способов получения информации о процессах. Один из основных классов, предоставляющих такую функциональность, - это **System.Diagnostics.Process**

* $"ID: {process.Id}\n" +
* $"Имя: {process.ProcessName}\n" +
* $"Приоритет: {process.PriorityClass}\n" +
* $"Время запуска: {process.StartTime}\n" +
* $"Текущее состояние(объем памяти, который выделен для данного процесса): {process.VirtualMemorySize64}\n" +
* $"Отвечает ли пользовательский интерфейс: {process.Responding}\n");

**3. Как создать и настроить домен?**

1. Создание и настройка домена приложения

AppDomain newDomain = AppDomain.CreateDomain("NewApplicationDomain");

newDomain.UnhandledException += NewDomain\_UnhandledException;

2.Загрузка и выполнение сборки в новом домене

newDomain.ExecuteAssembly("YourAssembly.exe");

3.Освобождение ресурсов домена приложения

AppDomain.Unload(newDomain);

**4. Как создать и настроить поток?**

1.Создание и запуск нового потока

Thread newThread = new Thread(MyThreadFunction);

newThread.Start();

**5. В каких состояниях может быть поток?**

Unstarted (Не начат): Поток находится в этом состоянии до вызова метода Thread.Start(). В этом состоянии поток не выполняет код.

Running (Выполняется): Поток находится в этом состоянии, когда он активен и выполняет свой код.

Blocked (Заблокирован): Поток может перейти в это состояние, когда он ожидает какого-то события, ресурса или выполнения условия. Например, при вызове метода Thread.Sleep(), Monitor.Wait(), ожидании завершения операции ввода-вывода и т.д.

Suspended (Приостановлен): Это состояние, в котором поток может быть временно остановлен. В .NET оно не используется широко, и рекомендуется использовать другие механизмы.

Aborted (Прерван): Поток может быть прерван с использованием метода Thread.Abort(). Это состояние, в котором поток завершает свою работу из-за вызова этого метода.

Stopped (Остановлен): Поток может находиться в этом состоянии после завершения своей работы (выхода из метода, который был запущен методом Thread.Start()).

**6. Какие методы управления потоками вы знаете, для чего и как их  
использовать?**

Thread.Sleep(int milliseconds): Приостанавливает выполнение текущего потока на указанное количество миллисекунд.

Thread.Join(): Блокирует текущий поток до тех пор, пока поток, на котором вызван метод, не завершится.

Thread.Abort(): Прерывает выполнение потока.

Thread.Suspend() и Thread.Resume(): методы для временной приостановки и возобновления выполнения потока.

Thread.Priority: Устанавливает или получает приоритет выполнения потока.

ManualResetEvent и AutoResetEvent: Средства синхронизации, предоставляющие механизмы для блокировки и разблокировки потоков.

**7. Какие приоритеты потока вы знаете?**

Highest (Самый высокий): Представляет самый высокий приоритет выполнения. Потоки с этим приоритетом получают больше ресурсов процессора в сравнении с потоками с более низким приоритетом.

AboveNormal (Выше среднего):Представляет приоритет, выше среднего. Потоки с этим приоритетом получают более высокий приоритет выполнения по сравнению с Normal.

Normal (Нормальный):Представляет стандартный приоритет выполнения. Это значение по умолчанию для новых потоков.

BelowNormal (Ниже среднего): Представляет приоритет, ниже среднего. Потоки с этим приоритетом получают более низкий приоритет выполнения.

Lowest (Самый низкий):Представляет самый низкий приоритет выполнения. Потоки с этим приоритетом получают наименьший доступ к ресурсам процессора.

**8. Что такое пул потоков и для чего он используется?**

Пул потоков (Thread Pool) представляет собой механизм в .NET, предназначенный для эффективного использования потоков. Он предоставляет пул готовых к использованию потоков, которые могут быть многократно использованы для выполнения различных задач. Вместо создания нового потока для каждой задачи при необходимости, потоки берутся из пула и возвращаются в него после завершения задачи.

**9. Что такое критическая секция? Поясните использование.**

Критическая секция — это участок кода, который может быть одновременно выполнен только одним потоком. Когда один поток входит в критическую секцию (защищённый участок кода), он блокирует другие потоки от входа в эту секцию до тех пор, пока первый поток не выйдет из неё. Это средство синхронизации используется для предотвращения одновременного доступа нескольких потоков к общим ресурсам, таким как переменные, объекты или участки кода, которые могут привести к непредсказуемым результатам или состояниям, если используются параллельно.

**10.Что такое мьютекс? Поясните использование**

Мьютекс (Mutex) — это средство синхронизации, предоставляемое операционной системой или библиотекой, которое используется для обеспечения взаимного исключения при доступе к общим ресурсам из нескольких потоков или процессов.

Принцип работы мьютекса прост: он может находиться в одном из двух состояний — "заблокирован" (захвачен) или "разблокирован" (освобожден). Только один поток или процесс может захватить мьютекс в определенный момент времени. Если другой поток или процесс пытается захватить мьютекс, когда он уже заблокирован, то он будет заблокирован до тех пор, пока мьютекс не будет освобожден текущим владельцем.

**11.Что такое семафор? Поясните использование**

Семафор — это средство синхронизации, используемое для управления доступом к общим ресурсам из нескольких потоков или процессов. Семафор обычно обеспечивает счетчик, который представляет собой количество разрешений на доступ к ресурсу. Когда поток или процесс хочет получить доступ к ресурсу, он уменьшает счетчик семафора, и если счетчик больше или равен нулю, доступ разрешается. Если счетчик равен нулю, поток блокируется до тех пор, пока не освободится хотя бы одно разрешение.

**12.Что такое неблокирующие средства синхронизации?**

Неблокирующие средства синхронизации — это подход к управлению доступом к общим ресурсам, при котором потоки не используют традиционные блокировки (например, мьютексы, семафоры) для синхронизации своей работы. Вместо этого они применяют алгоритмы, которые позволяют потокам обращаться к общим ресурсам параллельно, минимизируя при этом блокировку и конфликты между потоками.

Примеры неблокирующих структур данных включают "неблокирующий стек" (Lock-Free Stack) или "неблокирующую очередь" (Lock-Free Queue). Эти структуры данных обеспечивают доступ к элементам без использования блокировок, что может быть полезно в высокопроизводительных и многозадачных приложениях.

**13.Для чего можно использовать класс Timer?**

В .NET класс Timer из пространства имен System.Threading представляет собой механизм для выполнения кода через определенные промежутки времени. Он предоставляет возможность планирования выполнения метода делегата с определенной периодичностью.