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

**Процесс** – объект, который создаётся ОС для каждого приложения в момент его запуска. Характеризуется собственным адресным пространством, которое напрямую недоступно другим процессам.

**Домен** - полностью изолирует используемые в его рамках ресурсы (совместно используемые данные) как от других доменов того же самого процесса, так, естественно, от доменов приложения других процессов.

**Поток** - последовательность выполняемых команд процессора.

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

В .NET процесс представлен классом Process из пространства имен System.Diagnostics.

1. **Как создать и настроить домен?**

Домен создаётся использованием статического метода **AppDomain.CreateDomain().**

Загрузить сборку в домен можно с помощью метода **AppDomain.Load(),** с указанием имени сборки. Кроме того, существует метод **AppDomain.ExecuteAssembly(),** который позволяет загрузить сборку \*.ехе и выполнить метод Main().

С помощью метода **AppDomain.Unload()** можно производить избирательную выгрузку определенного домена приложения из обслуживающего процесса. В этом случае вместе с доменом приложения будут выгружаться и все содержащиеся в нем сборки.

1. **Как создать и настроить поток?**

Для создания нового потока используется делегат **ThreadStart**, который получает в качестве параметра метод. Чтобы запустить поток, вызывается метод **Start**. Метод **Sleep** позволяет приостановить поток на промежуток времени, указанный как параметр данного метода. Поток можно сделать фоновым, изменив значение свойсва **IsBackground**, изменить его приоритет **Priority**, изменить его имя **Name**.

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

* Выполнение;
* Ожидание - ожидания конца ввода-вывода;
* Готовность - готов работать, но компьютер занят другим.

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

Поток создается за счет вызова метода **Start()** объекта Thread. Однако после вызова метода Start() новый поток все еще пребывает не в состоянии **Running**, а в состоянии **Unstarted**. В состояние **Running** поток переходит сразу после того, как планировщик потоков операционной системы выберет его для выполнения. Информация о текущем состоянии потока доступна через свойство **Thread.ThreadState**.

С помощью метода **Thread.Sleep()** поток можно перевести в состояние **WaitSleepJoin** и при этом указать, через какой промежуток времени поток должен возобновить работу.

Чтобы остановить поток, необходимо вызвать метод **Thread.Abort().** При вызове этого метода в соответствующем потоке генерируется исключение типа **ThreadAbortException**. Чтобы продолжить выполнение потока после выдачи исключения **ThreadAbortException**, следует вызвать метод **Thread.ResetAbort().**

Если необходимо дожидаться завершения работы потока, можно вызвать метод **Thread.Join().** Этот метод блокирует текущий поток и переводит его в состояние **WaitSleepJoin** до тех пор, пока не будет завершен присоединенный к нему поток.

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

* Real time;
* Normal;
* High;
* Below normal;
* Above normal;
* Idle.

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

**Пул потоков** – автоматизированный ссылочный тип-контейнер ссылок на рабочие потоки на языке C#. Определен в пространстве имен библиотеки типов System. Treading. По сути выполняет роль посредника между планировщиком задач операционной системы и потоками, реализованными в рамках.

Если у вас много логических задач, требующих постоянной обработки, и вы хотите, чтобы это было сделано параллельно, используйте планировщик пула .

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

**Критическая секция** — участок исполняемого кода программы, в котором производится доступ к общему ресурсу (данным или устройству), который не должен быть одновременно использован более чем одним потоком выполнения. При нахождении в критической секции двух (или более) потоков возникает состояние «гонки» («состязания»).

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

Mutex является одним из классов в .NET Framework, позволяющих обеспечить синхронизацию среди множества процессов.

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

Еще один инструмент, для управления синхронизацией. Семафоры позволяют ограничить доступ определенным количеством объектов.

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

**Неблокирующая синхронизация** — подход в параллельном программировании на симметрично-многопроцессорных системах, в котором принят отказ от традиционных примитивов блокировки, таких, как семафоры, мьютексы и события. Разделение доступа между потоками идёт за счёт атомарных операций и специальных, разработанных под конкретную задачу, механизмов блокировки.

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

Данный класс позволяет запускать определенные действия по истечению некоторого периода времени.

Одна из перегрузок конструктора таймера принимает четыре параметра:

* объект делегата TimerCallback;
* объект, передаваемый в качестве параметра в метод делегата;
* количество миллисекунд, через которое таймер будет запускаться;
* интервал между вызовами метода делегата.

  // устанавливаем метод обратного вызова

         TimerCallback tm = new TimerCallback(Count);

         // создаем таймер

         Timer timer = new Timer(tm, num, 0, 2000);