1. Что такое поток?

Поток (или нитка выполнения) — это наименьшая единица выполнения в программе. Потоки являются частью процесса и делят общую память, что позволяет им обмениваться данными. Каждый поток имеет свой собственный стек и регистры, но работает с общей памятью и ресурсами процесса, что делает их более легковесными по сравнению с процессами. Потоки могут быть использованы для параллельной или многозадачной обработки, например, для выполнения нескольких задач одновременно.

2. Что такое Thread Local Storage (TLS)?

Thread Local Storage (TLS) — это механизм, который позволяет каждому потоку иметь свою собственную копию данных, которая недоступна другим потокам. Это полезно, когда нужно хранить уникальную информацию для каждого потока (например, идентификаторы, временные данные), избегая конфликтов между потоками при доступе к этим данным. В некоторых операционных системах и языках программирования поддерживается механизм TLS, позволяющий каждому потоку сохранять свои локальные переменные.

3. Когда лучше использовать потоки вместо дополнительных процессов для многозадачности?

Потоки лучше использовать, когда:

- Требуется более легковесная многозадачность, поскольку создание и переключение между потоками менее затратное по сравнению с процессами.

- Задачи выполняются в рамках одного приложения, и необходимо разделить вычисления или обработки, например, в веб-сервере, многозадачной игре и т. д.

- Нужно разделить общие ресурсы (например, память), так как потоки одного процесса могут обмениваться данными напрямую.

В то время как процессы могут быть предпочтительны, когда необходимо изолировать задачи (например, для обеспечения безопасности или стабильности), так как каждый процесс имеет собственную память.

4. Какие функции WinAPI для работы с потоками вы знаете? POSIX?

- WinAPI:

- `CreateThread`: Создает новый поток.

- `ExitThread`: Завершает выполнение потока.

- `WaitForSingleObject`: Ожидает завершения потока (или другого объекта синхронизации).

- `TerminateThread`: Принудительно завершает поток.

- `GetCurrentThread`: Получает дескриптор текущего потока.

- `ThreadLocalStorage`: Управление локальными переменными потока.

- POSIX:

- `pthread\_create`: Создает новый поток.

- `pthread\_exit`: Завершает выполнение потока.

- `pthread\_join`: Ожидает завершения потока.

- `pthread\_cancel`: Отправляет сигнал для отмены выполнения потока.

- `pthread\_self`: Получает идентификатор текущего потока.

- `pthread\_mutex\_lock`/`pthread\_mutex\_unlock`: Синхронизация потоков с помощью мьютексов.

5. К каким возможным последствиям может привести выполнение потоком следующего кода?

- `pthread\_join(pthread\_self(), NULL);`

Этот код приведет к взаимной блокировке (deadlock). Поток вызывает `pthread\_join` на себе, что означает ожидание завершения самого себя. Поток никогда не завершится, поскольку он ожидает, пока сам не завершится, а это невозможно. В результате программа зависнет, и поток не завершит свою работу.

- `WaitForSingleObject(GetCurrentThread(), INFINITE);`

Этот код также может привести к взаимной блокировке. Функция `WaitForSingleObject` ожидает завершения текущего потока (с использованием дескриптора потока `GetCurrentThread()`), а так как поток не завершится, выполнение программы застрянет в этом вызове. Поток будет ожидать самого себя, что приведет к зависанию.