1. Дайте определение понятию «синхронизация потоков».

Понятие "синхронизация потоков" относится к координации выполнения нескольких потоков в многопоточной программе. Оно описывает методы и механизмы, используемые для упорядочивания доступа к общим ресурсам или обмена информацией между потоками. Синхронизация потоков позволяет избежать гонок данных (race conditions) и других проблем, возникающих при параллельном выполнении кода.

1. Объясните понятие «взаимная блокировка».

"Взаимная блокировка" (deadlock) возникает, когда два или более потока находятся в состоянии ожидания ресурсов, которые контролируют другие потоки. Каждый из этих потоков блокирует ресурс, который требуется другому потоку для продолжения выполнения, при этом сам ожидает ресурс, контролируемый другим потоком. В результате ни один из потоков не может продолжить свое выполнение, поскольку он ожидает освобождения ресурса, которое не произойдет. Взаимная блокировка является нежелательным состоянием и может привести к замедлению или даже зависанию программы.

1. Перечислите механизмы авторизации OS.

Механизмы авторизации операционной системы (OS) включают в себя следующие методы:

- Парольная авторизация: Пользователь должен предоставить правильный пароль для доступа к системе или определенным ресурсам.

- Авторизация по ключу: Используется криптографический ключ для проверки подлинности и авторизации пользователя.

- Авторизация на основе сертификатов: Пользователь предоставляет цифровой сертификат, который содержит информацию о его подлинности, выпущенный доверенным удостоверяющим центром.

- Биометрическая авторизация: Используются физиологические или поведенческие характеристики пользователя, такие как отпечатки пальцев, сканирование сетчатки глаза или распознавание голоса.

- Авторизация на основе ролей: Пользователь получает доступ к определенным ресурсам в зависимости от своей роли или привилегий в системе.

1. Поясните в чем разница между механизмом **mutex** и **semaphore**.

Разница между механизмом mutex и semaphore заключается в их функциональности и способе использования:

- Mutex (мьютекс) применяется для обеспечения взаимного исключения, то есть только один поток может захватить мьютекс и получить доступ к защищаемому ресурсу. Остальные потоки будут ожидать освобождения мьютекса. Mutex может быть разблокирован только тем потоком, который его захватил.

- Semaphore (семафор) используется для контроля доступа к определенному количеству ресурсов. Семафор содержит счетчик, который отслеживает количество доступных ресурсов. Потоки могут захватывать и освобождать семафоры. Если счетчик семафора равен нулю, поток будет ожидать освобождения ресурса.

Основная разница между mutex и semaphore заключается в их использовании. Mutex применяется для обеспечения взаимного исключения в отношении конкретного ресурса, в то время как semaphore используется для контроля доступа к определенному количеству ресурсов.

1. Почему **mutex,** **semaphore, event** создают объект ядра OS, а **critical section** нет.

Mutex, semaphore и event создают объекты ядра операционной системы (OS) потому, что они предоставляют механизмы синхронизации, которые могут использоваться между разными процессами или потоками в рамках операционной системы. Эти объекты ядра предоставляют надежные и масштабируемые механизмы синхронизации, поддерживаемые операционной системой.

Mutex, semaphore и event могут быть использованы для синхронизации как внутри одного процесса, так и между разными процессами. Они могут быть именованными (named) или безымянными (unnamed). Именованные объекты могут быть открыты и использованы разными процессами, даже если они не являются родственными или не разделяют общую память. Безымянные объекты синхронизации обычно используются для синхронизации потоков внутри одного процесса.

С другой стороны, критическая секция (critical section) не создает объект ядра операционной системы. Критическая секция реализуется на уровне пользовательского кода и не требует взаимодействия с ядром операционной системы. Критическая секция представляет собой легковесный механизм синхронизации, который может использоваться только внутри одного процесса для синхронизации доступа к общим данным между потоками в этом процессе.

Поскольку критическая секция не требует взаимодействия с ядром операционной системы, она обычно работает быстрее и имеет меньшую накладную надежность по сравнению с объектами ядра, такими как mutex, semaphore и event. Однако, критическая секция имеет ограничения, например, она не может использоваться для синхронизации доступа между разными процессами.