1. **Синхронизация потоков** - это процесс управления и координации параллельных потоков выполнения в многозадачной или многопоточной среде с целью предотвращения конфликтов, гонок данных и других проблем, возникающих при одновременном доступе к общим ресурсам. Синхронизация потоков позволяет гарантировать правильное выполнение программы, особенно при одновременном доступе к разделяемым данным.
2. **Взаимная блокировка (deadlock)** - это ситуация, когда два или более потока блокируют друг друга, ожидая освобождения ресурсов, которые удерживают другие потоки. Это может произойти, когда потоки блокируют ресурсы, необходимые другим потокам, и ожидают освобождения ресурсов друг друга. В результате программы могут зависать и не продолжать выполнение.
3. **Механизмы авторизации OS** - это специальные методы и средства, используемые операционной системой для аутентификации пользователей и управления их доступом к ресурсам и функциям системы. Эти механизмы включают в себя пароли, аутентификацию посредством биометрических данных, ключи безопасности, системы прав доступа, а также системы авторизации на основе ролей и групп.
4. **Разница между mutex и semaphore**:

* **Mutex (мьютекс)** предназначен для обеспечения взаимного исключения во взаимодействии с ресурсами. Он позволяет только одному потоку одновременно захватить и освободить мьютекс.
* **Semaphore (семафор)** - это счетчик, который может контролировать доступ нескольких потоков к ресурсам. Семафор позволяет устанавливать ограничение на количество потоков, которые могут одновременно получить доступ к ресурсу.

1. **Mutex, semaphore, event** - это объекты синхронизации, которые создаются и управляются ядром операционной системы. Это означает, что они могут использоваться для синхронизации потоков и процессов, работающих в разных адресных пространствах и даже на разных уровнях привилегий. Они предоставляют мощные механизмы синхронизации, но их использование требует больше ресурсов и является более гибким.

**Critical section (критическая секция)** - это механизм синхронизации, специфичный для многопоточных программ, который создается и управляется пользовательским кодом, а не ядром ОС. Critical section обеспечивает взаимное исключение только между потоками в пределах одного процесса и работает быстрее, чем объекты синхронизации ядра, но ограничен только одним процессом. Critical section не требует доступа к ядру операционной системы, что делает его более эффективным в случае многопоточности в рамках одного процесса.

1. **Социальное время** — это время, в котором человеческая активность создает общество. [Оно характеризуется возможностью ускорения или замедления в зависимости от частоты событий1](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B2%D1%80%D0%B5%D0%BC%D1%8F). [Социальное время не монотонно, потому что оно может ускоряться или замедляться в зависимости от частоты событий и действий людей1](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B2%D1%80%D0%B5%D0%BC%D1%8F).
2. **Эпоха Linux** — это начальная точка отсчета времени в Unix-подобных операционных системах, таких как Linux. [Эта эпоха начинается с 00:00:00 UTC 1 января 1970 года2](https://habr.com/ru/articles/695688/). [Время в этих системах обычно измеряется в секундах, прошедших с начала эпохи2](https://habr.com/ru/articles/695688/).
3. **Universal Coordinated Time (UCT)** или Скоординированное всемирное время — это основной стандарт времени, по которому мир регулирует часы и время. [Он находится примерно в пределах одной секунды от среднего солнечного времени (такого как UT1) на 0° долготы (на IERS Reference Meridian как текущий используемый первый меридиан) и не корректируется на летнее время3](https://en.wikipedia.org/wiki/Coordinated_Universal_Time).
4. **Относительное время** и **абсолютное время** — это два разных способа представления времени. Абсолютное время обычно относится к фиксированному отсчету времени, такому как дата или время суток. [Относительное время, с другой стороны, обычно относится к продолжительности или интервалу времени, который прошел от определенного события4](https://fb.ru/article/468906/absolyutnoe-vremya-ponyatie-osnovyi-teorii).
5. В контексте информатики, **тик** обычно относится к самому короткому интервалу времени, который может измерить система. Это может быть основано на частоте процессора или на других внутренних таймерах системы.
6. **Ожидающий таймер** — это таймер, который был запущен и ожидает истечения определенного интервала времени. В Linux есть несколько типов таймеров, включая одноразовые таймеры (срабатывают один раз после истечения определенного интервала времени), периодические таймеры (срабатывают через регулярные интервалы времени) и таймеры высокого разрешения (для более точного измерения времени). [Таймер может находиться в одном из нескольких состояний, включая активное (ожидание истечения времени), остановленное (не активно и не ожидает истечения времени) и истекшее (время истекло и таймер сработал)5](https://learn.microsoft.com/ru-ru/dotnet/standard/threading/timers).
7. В Linux используются различные типы часов для различных целей. Например, системные часы используются для отслеживания текущего времени, а монотонные часы используются для измерения прошедшего времени. [Кроме того, есть процессорные часы, которые измеряют время, затраченное на выполнение процесса6](https://habr.com/ru/companies/vdsina/articles/505314/).
8. **HZ** и **CLOCKS\_PER\_SEC** — это две константы, используемые в системах Unix и Linux. HZ обычно определяет количество тиков системного таймера в секунду и зависит от конкретной системы. [CLOCKS\_PER\_SEC определяет количество тактов системных часов в секунду и используется для пересчета величины, возвращаемой функцией clock(), в секунды7](https://ru.wikipedia.org/wiki/Time.h).