**1. Что такое межпроцессное взаимодействие (IPC)?**

**Межпроцессное взаимодействие (Inter-Process Communication, IPC)** — это набор механизмов, позволяющий процессам обмениваться данными, синхронизировать свои действия и совместно использовать ресурсы. Эти механизмы используются в операционных системах для обеспечения взаимодействия между разными процессами, которые могут работать как в одном адресном пространстве, так и в разных.

**2. Какие группы IPC-механизмов вам известны? Какие механизмы входят в эти группы?**

IPC-механизмы можно разделить на несколько групп:

**1. Механизмы обмена сообщениями:**

* **Очереди сообщений** (Message Queues).
* **Каналы (Pipes):**
  + Анонимные каналы (Anonymous Pipes).
  + Именованные каналы (Named Pipes).
* **Сокеты** (Sockets).
* **Очереди сообщений ZeroMQ, Kafka** (продвинутые брокеры сообщений).

**2. Механизмы совместного использования памяти:**

* **Разделяемая память (Shared Memory).**
* **Память, отображённая на файл (Memory-Mapped Files).**

**3. Механизмы синхронизации:**

* **Мьютексы** (Mutexes).
* **Семафоры** (Semaphores).
* **События** (Events).
* **Критические секции** (Critical Sections).
* **Барьеры** (Barriers).

**4. Высокоуровневые механизмы:**

* **RPC (Remote Procedure Call).**
* **CORBA, gRPC.**

**3. Какой механизм взаимодействия выбрать, когда требуется максимальная скорость передачи данных?**

Для максимальной скорости передачи данных оптимально использовать **разделяемую память (Shared Memory)**. Этот механизм позволяет процессам совместно использовать один и тот же участок памяти без необходимости копирования данных, что минимизирует накладные расходы.

**4. Какой механизм взаимодействия выбрать, когда требуется повышенная устойчивость к ошибкам?**

Для обеспечения повышенной устойчивости к ошибкам лучше выбрать **очереди сообщений** или **сокеты**, поскольку:

* Эти механизмы могут гарантировать доставку сообщений даже при сбоях, если данные кэшируются.
* Например, системы брокеров сообщений (Kafka, RabbitMQ) обеспечивают надёжность с помощью журналирования.

**5. Что такое синхронизация?**

**Синхронизация** — это процесс упорядочивания действий нескольких потоков или процессов для обеспечения их корректного взаимодействия. Синхронизация гарантирует:

* Последовательный доступ к разделяемым ресурсам.
* Отсутствие состояния гонки (race condition).
* Согласованность данных.

**6. Какие механизмы синхронизации вы знаете? Объясните любые три механизма.**

**Известные механизмы:**

* **Мьютексы** (Mutexes).
* **Семафоры** (Semaphores).
* **События** (Events).
* **Критические секции** (Critical Sections).
* **RWLock (Reader-Writer Locks).**

**1. Мьютекс:**

Мьютекс — это механизм, позволяющий одному потоку или процессу эксклюзивно владеть ресурсом. Если ресурс занят, другие потоки блокируются до освобождения мьютекса.

* **Плюс:** Простота в использовании.
* **Минус:** Возможность взаимной блокировки (deadlock).

**2. Семафор:**

Семафор управляет доступом к ресурсу через счётчик. Если счётчик больше нуля, доступ разрешается; если равно нулю — поток блокируется.

* **Плюс:** Может использоваться для ограничения числа потоков, одновременно использующих ресурс.
* **Минус:** Более сложный в настройке, чем мьютекс.

**3. Событие:**

Событие — это механизм для синхронизации потоков, который позволяет одному потоку ожидать наступления определённого условия, инициируемого другим потоком.

* **Плюс:** Удобен для уведомления нескольких потоков о состоянии.
* **Минус:** Требует явного управления состоянием события.

**7. Какое главное отличие CriticalSection в Windows от всех остальных механизмов? Какой основной плюс и минус этого отличия?**

**Отличие:**

**CriticalSection** работает только внутри одного процесса и основана на использовании атомарных операций. В отличие от других механизмов (например, мьютексов), она не требует переключения контекста ядра при нормальных условиях.

**Основной плюс:**

* **Высокая производительность**: Благодаря работе в пространстве пользователя, переключение в режим ядра чаще всего не требуется.

**Основной минус:**

* **Ограничение области применения**: CriticalSection нельзя использовать для синхронизации между процессами, только между потоками одного процесса.