Процессам часто нужно взаимодействовать друг с другом, например, один процесс может передавать данные другому процессу, или несколько процессов могут обрабатывать данные из общего файла. Во всех этих случаях возникает проблема синхронизации процессов, которая может решаться приостановкой и активизацией процессов, организацией очередей, блокированием и освобождением ресурсов.

Задачи межпроцессного взаимодействия:
1. обмен информацией между процессами,
2. недопущение конфликтных ситуаций,
3. согласование действий процессов.

Пренебрежение вопросами синхронизации процессов, выполняющихся в режиме мультипрограммирования, может привести к их неправильной работе или даже к краху системы.

Синхронизация нужна для того, чтобы избежать:

- *Гонок* (или взаимных состязаний) - возникают когда 2 или более процессов обрабатывают разделяемые данные и конечный результат зависит от соотношения скоростей потоков.
- *Тупиков* (взаимных блокировок, клинчей) - возникает когда два взаимосвязанных процесса блокируют друг друга при обращении к критической области.

Обмен информацией между процессами
Средства обмена информацией можно разделить на три категории:
Сигнальные. Передается минимальное количество информации – один бит, "да" или "нет". Используются, как правило, для извещения процесса о наступлении какого-либо события.

Канальные. «Общение» процессов происходит через линии связи, предоставленные операционной системой, и напоминает общение людей по телефону. Объем передаваемой информации в единицу времени ограничен пропускной способностью линий связи.

Разделяемая память. Два или более процессов могут совместно использовать некоторую область адресного пространства. Созданием разделяемой памяти занимается операционная система (если, конечно, ее об этом попросят). «Общение» процессов напоминает совместное проживание студентов в одной комнате общежития. Возможность обмена информацией максимальна, как, впрочем, и влияние на поведение другого процесса, но требует повышенной осторожности (если вы переложили на другое место вещи вашего соседа по комнате, а часть из них еще и выбросили). Использование разделяемой памяти для передачи/получения информации осуществляется с помощью средств обычных языков программирования, в то время как сигнальным и канальным средствам коммуникации для этого необходимы специальные системные вызовы. Разделяемая память представляет собой наиболее быстрый способ взаимодействия процессов в одной вычислительной системе.

В случае конкурирующих процессов (потоков) возможно возникновение трех проблем. Первая из них – необходимость **взаимных исключений** (mutual exclusion).
Осуществление взаимных исключений создает две дополнительные проблемы. Одна из них – **взаимоблокировки** (deadlock) или **тупики**. Рассмотрим, например, два процесса – P1 и P2, и два ресурса – R1 и R2. Предположим, что каждому процессу для выполнения части своих функций требуется доступ к общим ресурсам. Тогда возможно возникновение следующей ситуации: ОС выделяет ресурс R1 процессу P2, а ресурс R2 – процессу P1. В результате каждый процесс ожидает получения одного из двух ресурсов. При этом ни один из них не освобождает уже имеющийся ресурс, ожидая получения второго ресурса для выполнения функций, требующих наличие двух ресурсов. В результате процессы оказываются взаимно заблокированными.