Довольно часто процессам необходимо взаимодействовать с другими процессами. Например, в канале оболочки выходные данные одного процесса могут передаваться другому процессу, и так далее вниз по цепочке. Поэтому возникает необходимость во взаимодействии процессов, и желательно, по хорошо продуманной структуре без использования прерываний.

Способы взаимодействия процессов (потоков) можно классифицировать по степени осведомленности одного процесса о существовании другого.

1. Процессы ***не осведомлены*** о наличии друг друга (например, процессы разных заданий одного или различных пользователей). Это независимые процессы, не предназначенные для совместной работы. Хотя эти процессы и не работают совместно, ОС должна решать вопросы конкурентного использования ресурсов. Например, два независимых приложения могут затребовать доступ к одному и тому же диску или принтеру. ОС должна регулировать такие обращения.
2. Процессы ***косвенно осведомлены*** о наличии друг друга (например, процессы одного задания). Эти процессы не обязательно должны быть осведомлены о наличии друг друга с точностью до идентификатора процесса, однако они разделяют доступ к некоторому объекту, например, буферу ввода-вывода, файлу или БД. Такие процессы демонстрируют сотрудничество при разделении общего объекта.
3. Процессы ***непосредственно осведомлены*** о наличии друг друга (например, процессы, работающие последовательно или поочередно в рамках одного задания). Такие процессы способны общаться один с другим с использованием идентификаторов процессов и изначально созданы для совместной работы. Эти процессы также демонстрируют сотрудничество при работе.

Таким образом, потенциальные проблемы, связанные с взаимодействием и синхронизацией процессов и потоков, могут быть представлены следующей таблицей.

Степень осведомленности	**Взаимосвязь**	**Влияние одного процесса на другой**	**Потенциальные проблемы**
Процессы не осведомлены друг о друге	Конкуренция	Результат работы одного процесса не зависит от действий других. Возможно влияние одного процесса на время работы другого	Взаимоисключения Взаимоблокировки Голодание
Процессы косвенно осведомлены о наличии друг друга	Сотрудничество с использованием разделения	Результат работы одного процесса может зависеть от информации, полученной от других. Возможно влияние одного	Взаимоисключения Взаимоблокировки Голодание Синхронизация
Процессы непосредственно осведомлены о наличии друг друга	Сотрудничество с использованием связи	Результат работы одного процесса зависит от информации, полученной от других процессов. Возможно влияние	Взаимоблокировки (расходуемые ресурсы) Голодание

одного процесса на
время работы
другого

При необходимости использовать один и тот же ресурс параллельные процессы вступают в конфликт (конкурируют) друг с другом. Каждый из процессов не подозревает о наличии остальных и не повергается никакому воздействию с их стороны. Отсюда следует, что каждый процесс не должен изменять состояние любого ресурса, с которым он работает. Примерами таких ресурсов могут быть устройства ввода-вывода, память, процессорное время, часы.

Между конкурирующими процессами не происходит никакого обмена информацией. Однако выполнение одного процесса может повлиять на поведение конкурирующего процесса. Это может, например, выразиться в замедлении работы одного процесса, если ОС выделит ресурс другому процессу, поскольку первый процесс будит ждать завершения работы с этим ресурсом. В предельном случае блокированный процесс может никогда не получить доступ к нужному ресурсу и, следовательно, никогда не сможет завершиться.

В случае конкурирующих процессов (потоков) возможно возникновение трех проблем. Первая из них – необходимость взаимных исключений (mutual exclusion). Предположим, что два или большее количество процессов требуют доступ к одному неразделяемому ресурсу, как например принтер. О таком ресурсе будем говорить как о критическом ресурсе, а о части программы, которая его использует, – как о критическом разделе (critical section) программы. Крайне важно, чтобы в критической ситуации в любой момент могла находиться только одна программа. Например, во время печати файла требуется, чтобы отдельный процесс имел полный контроль над принтером, иначе на бумаге можно получить чередование строк двух файлов.