Синхронизация процессов и потоков.

В многозадачной ОС синхронизация процессов и потоков необходима для исключения конфликтных ситуаций при обмене данными между ними, разделении данных, доступе к процессору и устройствам ввода-вывода. Во многих ОС эти средства называются средствами межпроцессного взаимодействия (InterProcess Communications – IPC). Выполнение потока в многозадачной среде имеет асинхронный характер, поэтому сложно сказать, на каком этапе выполнения будет находиться поток в определенный момент времени. Задача синхронизации заключается в согласовании скоростей потоков путем их приостановки до наступления некоторого события и последующей активизации при наступлении этого события. Пренебрежение вопросами синхронизации процессов, выполняющихся в многозадачной системе, может привести к неправильной их работе или даже к краху системы. Рассмотрим в качестве примера программу печати файлов (принт-сервер). Эта программа печатает по очереди все файлы, имена которых последовательно в порядке поступления записывают в специальный общедоступный файл "заказов" другие программы. В данном случае это процессы-клиенты R и S, содержащие операции R1, R2, R3 и S1, S2, S3 (рис. 1). Особая переменная NEXT, также доступная всем процессам-клиентам, содержит номер первой свободной для записи имени файла позиции файла "заказов". Процессы-клиенты читают эту переменную, записывают в соответствующую позицию файла "заказов" имя своего файла и наращивают значение NEXT на единицу. Предположим, что в некоторый момент процесс R решил распечатать свой файл, для этого он прочитал значение переменной NEXT, значение которой предположим равно 4. Процесс запомнил это значение, но поместить имя файла не успел, так как его выполнение было прервано (например, вследствие исчерпания кванта). Очередной процесс S, желающий распечатать файл, прочитал то же самое значение переменной NEXT, поместил в четвертую позицию имя своего файла и нарастил значение переменной на единицу. Когда в очередной раз управление будет передано процессу R, то он, продолжая свое выполнение, в полном соответствии со значением текущей свободной позиции, полученным во время предыдущей итерации, запишет имя файла также в позицию 4, поверх имени файла процесса S. Таким образом, файл процесса S не будет напечатан.

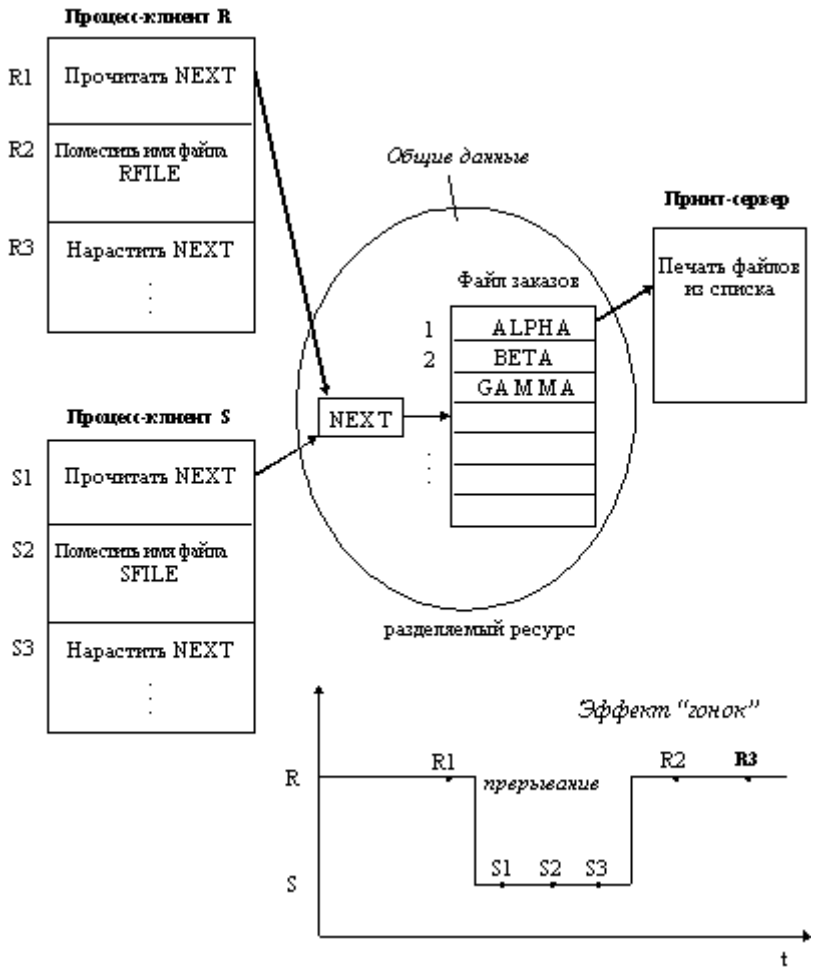

Рис. 1 Доступ процессов к разделяемым данным

Можно представить и другое развитие событий: были потеряны файлы нескольких процессов или, напротив, не был потерян ни один файл. В данном случае все определяется взаимными скоростями процессов и моментами их прерывания. Сложность проблемы синхронизации состоит в нерегулярности возникающих ситуаций. Ситуация, когда два или более процессов обрабатывают разделяемые данные, и конечный результат зависит от соотношения скоростей процессов, называется состоянием состязания или гонками.

Критическая секция.

Важным понятием синхронизации потоков является понятие «критическая секция». Критическая секция – это часть программы, которая должна выполняться без прерываний со стороны других потоков. Критическая секция всегда определяется по отношению к определенным критическим данным, при несогласованном изменении которых результат выполнения программы может быть непредсказуем. Во всех потоках, работающих с критическими данными, должна быть определена критическая секция, которая в общем случае состоит из разных последовательностей команд.

Критическая секция, например, используется для предоставления доступа к разделяемым ресурсам ВС. Чтобы исключить эффект гонок по отношению к разделяемым данным, необходимо обеспечить, чтобы в каждый момент в критической секции, связанной с этими данными, находился только один поток. Этот прием называют взаимным исключением. При этом неважно, находится этот поток в активном или в приостановленном состоянии. Для реализации взаимных исключений используются различные способы: запрещение прерываний, блокирующие переменные, семафоры, синхронизирующие объекты ОС.