В литературе по операционным системам можно встретить множество интересных проблем использования различных методов синхронизации, ставших предметом широких дискуссий и анализа.

Задача обедающих философов

В 1965 году Дейкстра сформулировал и решил проблему синхронизации, названную им задачей обедающих философов. С тех пор все изобретатели очередного примитива синхронизации считали своим долгом продемонстрировать его наилучшие качества, показав, насколько элегантно с его помощью решается задача обедающих философов.

Суть задачи довольно проста. Пять философов сидят за круглым столом и у каждого из них есть тарелка спагетти. Эти спагетти настолько скользкие, что есть их можно только двумя вилками. Между каждыми двумя тарелками лежит одна вилка. Жизнь философа состоит из чередующихся периодов приема пищи и размышлений. (Это положение из разряда абстракций даже в отношении философов, но вся их остальная деятельность к задаче не относится.) Когда философ становится голоден, он старается поочередно в любом порядке завладеть правой и левой вилкой. Если ему удастся взять две вилки, он на некоторое время приступает к еде, затем кладет обе вилки на стол и продолжает размышления. Основной вопрос состоит в следующем: можно ли написать программу для каждого философа, который действует предполагаемым образом и никогда при этом не попадает в состояние зависания? Процедура take_fork ждет, пока вилка не освободится, и берет ее. К сожалению, это решение ошибочно. Допустим, что все пять философов одновременно берут левую от себя вилку. Тогда никто из них не сможет взять правую вилку, что приведет к взаимной блокировке.
Можно изменить программу так, чтобы после получения левой вилки программа проверяла доступность правой вилки. Если эта вилка недоступна, философ кладет на место левую вилку, ожидает какое-то время, а затем повторяет весь процесс. Это предложение также ошибочно, но уже по другой причине. При некоторой доле невезения все философы могут приступить к выполнению алгоритма одновременно, взяв левые вилки и увидев, что правые вилки недоступны, опять одновременно положить на место левые вилки, и так до бесконечности. Подобная ситуация, при которой все программы бесконечно работают, но не могут добиться никакого прогресса, называется голоданием, или зависанием процесса.

Задача читателей и писателей

Задача обедающих философов хороша для моделирования процессов, которые соревнуются за исключительный доступ к ограниченному количеству ресурсов, например к устройствам ввода-вывода. Другая общеизвестная задача касается читателей и писателей. В ней моделируется доступ к базе данных. Представим, к примеру, систему бронирования авиабилетов, в которой есть множество соревнующихся процессов, желающих обратиться к ней по чтению и записи. Вполне допустимо наличие нескольких процессов, одновременно считывающих информацию из базы данных, но если один процесс обновляет базу данных (проводит операцию записи), никакой другой процесс не может получить доступ к базе данных даже для чтения информации. Вопрос в том, как создать программу для читателей и писателей?

В этом решении первый читатель для получения доступа к базе данных выполняет в отношении семафора db операцию down. А все следующие читатели просто увеличивают значение счетчика rc. Как только читатели прекращают свою работу, они уменьшают значение счетчика, а последний из них выполняет в отношении семафора операцию up, позволяя заблокированному писателю, если таковой имеется, приступить к работе. В представленном здесь решении есть одна достойная упоминания недостаточно очевидная особенность. Допустим, что какой-то читатель уже использует базу данных, и тут появляется еще один читатель. Поскольку одновременная работа двух читателей разрешена, второй читатель допускается к базе данных. По мере появления к ней могут быть допущены и другие дополнительные читатели.