

Взаимное исключение

Мы продолжим изучать процессы, в частности рассмотрим такую ситуацию, когда у нас есть много процессов, но при этом процессоров, которые эти процессы выполняют – тоже много. Это с одной стороны хорошо, это то, к чему мы стремились (помните, мы начинали с того, что у нас один процессор, который очень быстрый, а всё остальное медленное, но нам нужно обработать много процессов, и поэтому придется переключаться между ними, чтобы сделать симуляцию многозадачности, и кажется, что добавив несколько процессоров или ядер процессора мы решим эту задачу, потому что теперь не нужно ждать, не нужно переключаться – мы реально можем делать что-то параллельно, и отчасти это верно), но любое усложнение, любое добавление новых возможностей – усложняет всё. Многопроцессорность и параллелизация, естественно, усложняет процесс и, так как всем этим чаще всего управляет именно операционная система, то на долю операционной системы выпадает очень большая ответственность по разрешению всех возможных проблем.

Когда, допустим, два человека живут в одной квартире (по большей части эти люди могут делать что-то параллельно по всей квартире: кто-то сидит в одном месте и что-то делает, кто-то в другом что-нибудь читает), то такая комната как туалет обычно используется одним человеком в один момент времени. Если кто-то туда зашел, он заперся и правило очень простое: если там кто-то находится – ты должен ждать, ты не можешь зайти пока там кто-то есть. Иными словами, туалетом в один момент времени может пользоваться только один человек. Это очень похоже на классическую ситуацию в мире компьютеров и операционных систем, когда вместо туалета есть какой-то ресурс и ресурс по своей природе таков, что им может пользоваться в один момент только один процесс. Если несколько процессов хотят им пользоваться, то нужна какая-то очередь, нужна какая-то организация всего этого, чтобы процессы не пытались туда залезть одновременно и не поломали этот виртуальный туалет. Таким образом **проблема решается с помощью замка. Если закрыто – жди.**

Свойства системы

В целом свойства системы таковы: у нас есть **несколько агентов** (в нашем случае это несколько процессов, но это также могут быть несколько отдельных программ или несколько отдельных потоков или treadов – неважно, главное, что это несколько процессов в общем плане, то есть несколько протекающих процессов, которые в какой-то момент времени захотят использовать какой-то ресурс). Эти ресурсы мы будем называть общими, потому что несколько процессов хотят ими пользоваться и это нужно как-то ограничить – это **общие ресурсы**. Ну и нужны какие-то **правила доступа к ним** (что-то вроде правила туалета о котором все знают, или, скажем, правила дорожного движения – тоже хороший пример синхронизации и параллелизации множества машин, которые хотят куда-то попасть: все хотят проехать перекресток, но одновременно два потока машин, допустим, не могут проезжать перекрестки, поэтому есть светофор).

Несколько процессов

У нас есть ситуация с несколькими процессами и **центральная проблема (или задача^{*}) современных операционных систем** – это то, как справиться с этими процессами, как справиться с большим количеством процессов, которые все чего-то хотят, всё нужно обслужить, при этом **ресурсов ограниченное количество** и многие из этих ресурсов общие, и многие из этих ресурсов **не могут одновременно использоваться несколькими процессами** (при этом многие ресурсы могут одновременно использоваться несколькими процессами), поэтому нужно создать такие правила и такие условия для этих процессов, чтобы в целом всё было наиболее оптимально.

* Задача или проблема – это свойство слова «проблема» в английском языке. Как правило проблемами (в особенности в образовании или на лекциях, на курсах) называют задачи. Проблема, это не что-то, что случилось и нужно исправить, проблема – это некая задача, которую нужно решить. С точки зрения русского языка, это может быть не совсем верно, лучше говорить – «задачи», однозначно будет понятно, что эта задача, но иногда в тексте будет встречаться слово «проблема». Имейте ввиду, что проблема – это задача (и естественно эта проблема, естественно это что-то, что не совсем хорошо и что нужно решить), но здесь суть именно в том, что мы можем это решить, мы можем подойти к этой задаче как-то, мы можем что-то придумать и решить эту проблему.

Когда у нас есть много процессов, когда у нас есть много элементов, которые эти процессы могут одновременно исполнять, когда процессы выполняются одновременно – **нам нужно управлять взаимодействием нескольких процессов**, и операционная система является этим очень важным агентом, который занимается этим.

Ситуация, которую мы видели раньше – это вроде как почти параллельное программирование, которое на самом деле совершенно не параллельно.

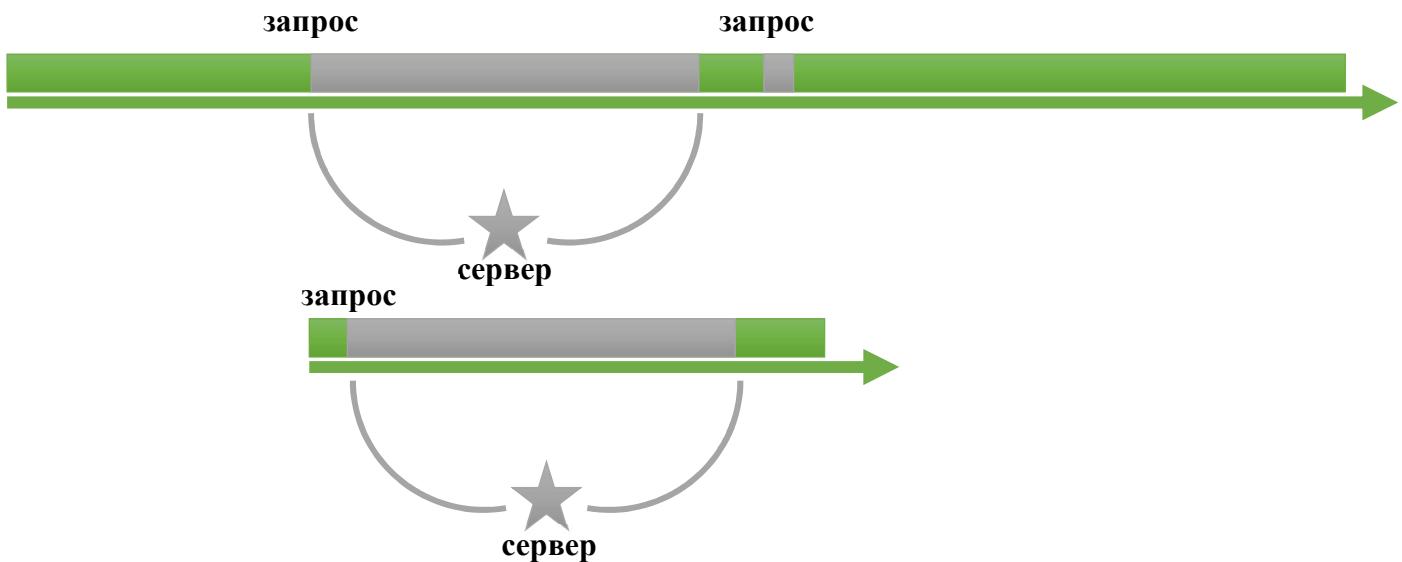


Рисунок 1. Не параллельное программирование

На рисунке 1 мы видим два процесса (в частности то, что сверху – это процесс, а внизу – это его тренд, его дополнительный поток) и мы делали два запроса к серверу, и как бы почти это было одновременно. На самом деле это было не одновременно, потому что мы в этот момент ждали и ничего не делали.

В параллельном программировании ситуация похожая (рисунок 2), но мы на самом деле реально делаем что-то одновременно.

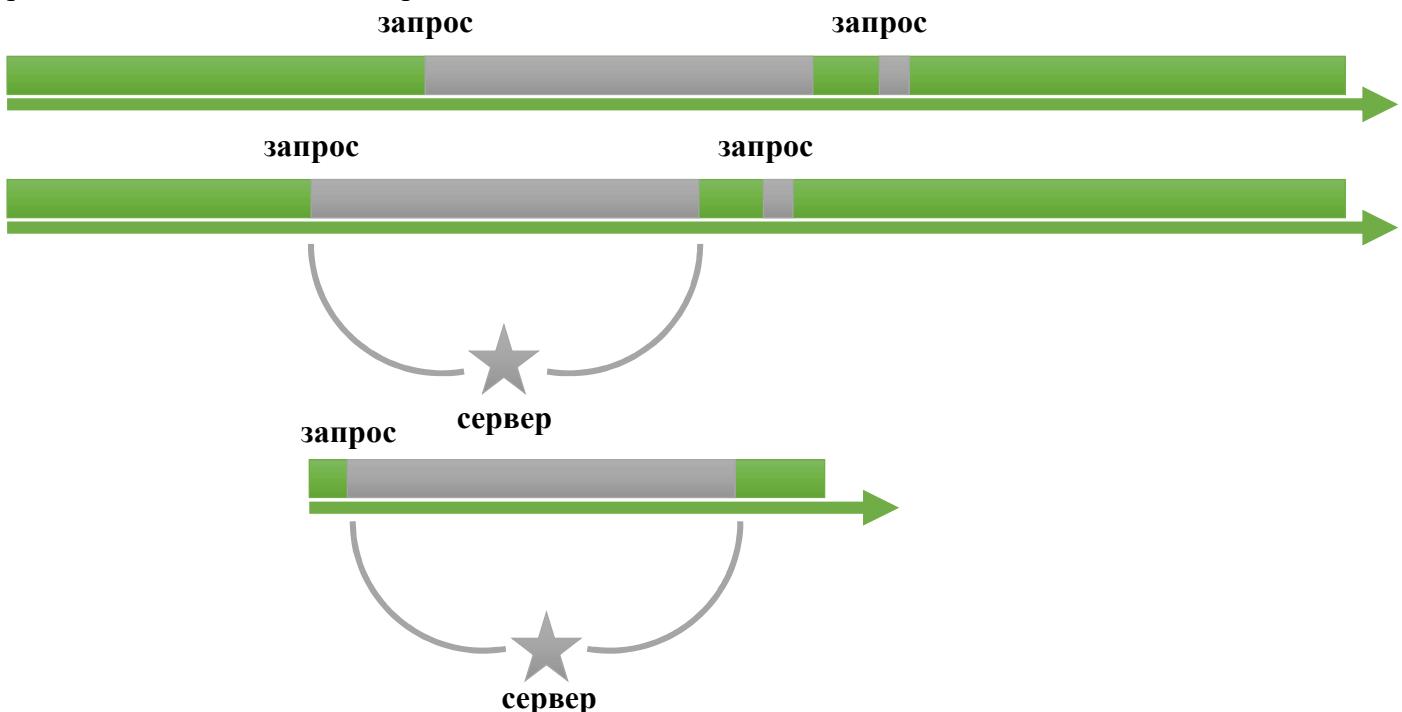


Рисунок 2. Параллельное программирование

На рисунке 2 у нас есть два потока и суть в том, что мы реально можем одновременно выполнять какие-то два процесса, но сразу же стоит немножко расстроиться, потому что из таких условий вытекает большое количество проблем.