ПОНЯТИЯ И ПРИМЕРЫ СОЗТЯЗАТЕЛЬНОЙ СИТУАЦИИ И КРИТИЧЕСКОЙ ОБЛАСТИ

Ситуация, когда два или более процесса считывают или записывают какие-нибудь общие данные, а окончательный результат зависит от того, какой процесс и когда именно выполняется, называется **состязатель- ной ситуацией**.

Чтобы посмотреть, как взаимодействие процессов осуществляется на практике, давайте рассмотрим простой общеизвестный пример — спулер печати. Когда процессу необходимо распечатать какой-нибудь файл, он помещает имя этого файла в специальный **каталог спулера**.

Другой процесс под названием **демон принтера** периодически проверяет наличие фай- лов для печати и в том случае, если такие файлы имеются, распечатывает их и удаляет их имена из каталога.

Представьте, что в нашем каталоге спулера имеется большое количество областей памяти с номерами 0, 1, 2..., в каждой из которых может храниться имя файла. Также представьте, что есть две общие переменные: *out*, указывающая на следующий файл, предназначенный для печати, и *in*, указывающая на следующую свободную область в каталоге. Эти две переменные могли бы неплохо сохраняться в файле, состоящем из двух слов и доступном всем процессам. В какой-то момент времени области от 0 до 3 пустуют (файлы уже распечатаны). Почти одновременно процессы *А* и *Б* решают, что им нужно поставить файл в очередь на печать.

КРИТИЧЕСКИЕ ОБЛАСТИ

Как же избежать состязательной ситуации? Ключом к предупреждению проблемы в этой и во многих других ситуациях использования общей памяти, общих файлов и вообще чего-нибудь общего может послужить определение способа, при котором в каждый конкретный момент времени доступ к общим данным для чтения и записи может получить только один процесс. Иными словами, нам нужен способ **взаимного исключения**, то есть некий способ, обеспечивающий правило, при котором если общие данные или файл используются одним процессом, возможность их использования всеми другими процессами исключается.

Проблемы обхода состязательных ситуаций могут быть сформулированы также в абстрактной форме. Какую-то часть времени процесс занят внутренними вычис- лениями и чем-нибудь другим, не создающим состязательных ситуаций. Но иногда он вынужден обращаться к общей памяти или файлам либо совершать какие-нибудь другие значимые действия, приводящие к состязаниям. Та часть программы, в которой используется доступ к общей памяти, называется **критической областью** или **крити- ческой секцией**. Если бы удалось все выстроить таким образом, чтобы никакие два процесса не находились одновременно в своих критических областях, это позволило бы избежать состязаний.

Хотя выполнение этого требования позволяет избежать состязательных ситуаций, его недостаточно для того, чтобы параллельные процессы правильно выстраивали совмест- ную работу и эффективно использовали общие данные. Для приемлемого решения необходимо соблюдение четырех условий:

1. Два процесса не могут одновременно находиться в своих критических областях.
2. Не должны выстраиваться никакие предположения по поводу скорости или ко- личества центральных процессоров.
3. Никакие процессы, выполняемые за пределами своих критических областей, не могут блокироваться любым другим процессом.
4. Процессы не должны находиться в вечном ожидании входа в свои критические области.