МЕТОДЫ ВЗАИМНОГО ИСКЛЮЧЕНИЯ С АКТИВНЫМ ОЖИДАНИЕМ

Взаимное исключение - некий способ, обеспечивающий правило, при котором если общие данные или файл используются одним процессом, возможность их использования всеми другими процессами исключается.

**Запрещение прерываний**

В однопроцессорных системах простейшим решением является запрещение всех пре- рываний каждым процессом сразу после входа в критическую область и их разрешение сразу же после выхода из критической области. При запрещении прерываний не могут осуществляться никакие прерывания по таймеру. Поскольку центральный процессор переключается с одного процесса на другой в результате таймерных или каких-нибудь других прерываний, то при выключенных прерываниях он не сможет переключиться на другой процесс. Поскольку процесс запретил прерывания, он может исследовать и обновлять общую память, не опасаясь вмешательства со стороны любого другого процесса.

**Блокирующие переменные**

В качестве второй попытки рассмотрим программное решение, в котором используется одна общая (блокирующая) переменная, исходное значение которой равно нулю. Когда процессу требуется войти в свою критическую область, сначала он проверяет значение блокирующей переменной. Если оно равно 0, процесс устанавливает его в 1 и входит в критическую область. Если значение уже равно 1, процесс просто ждет, пока оно не станет равно нулю. Таким образом, нулевое значение показывает, что ни один из про- цессов не находится в своей критической области, а единица — что какой-то процесс находится в своей критической области.

Постоянная проверка значения переменной, пока она не приобретет какое-нибудь значение, называется **активным ожиданием**. Как правило, этого ожидания следует избегать, поскольку оно тратит впустую время центрального процессора. Активное ожидание используется только в том случае, если есть основание полагать, что оно будет недолгим. Блокировка, использующая активное ожидание, называется **спин-блокировкой**.

**Строгое чередование**

Фактически это решение требует, чтобы, к примеру, при помещении файлов в каталог спулера два процесса входили в свои критические области, строго чередуясь друг с дру- гом. Ни одному из них не разрешено поместить файл в спулер два раза подряд. Хотя этот алгоритм позволяет предотвращать любые состязательные ситуации, его нельзя рассматривать в качестве серьезного кандидата на решение проблемы, поскольку он нарушает третье условие.

**Алгоритм Петерсона**

**Команда TSL**