Взаимодействие параллельных процессов

Достоинство семафоров: устранение активного ожидания на процессоре (когда процессор тратит квант процессорного времени на проверку флага или переменной).

Недостаток: переход в режим ядра (только ядро может заблокировать или разблокировать процесс).

!!! В ЛР необходимо при реализации создавать набор из трех семафоров.

Для решения проблемы структурирования была придумана концепция **мониторов**.

Мониторы

Мониторы были разработаны как средства более высокого уровня, чем т.н. примитивы ядра (примитив - не значит глупый, а значит низкоуровневое средство в распоряжении процессов).

Расмотренные выше команды test-and-set для разных систем можно объединить под общими названиями **lock** и **unlock**:

```
lock() – unlock()
```

```
wait() – post()
wait() – signal
```

Монитор - механизм, унифицирующий управление взаимоисключением (???)

Монитор обозначается ключевым словом monitor, при этом монитор может предоставляться операционной системой или ЯП.

Монитор - это набор процедур и данных, причем обращаться к данным монитора можно обращаться только с помощью процедур. Процесс, вызвавший процедуру монитора называется **процессом, находящимся в мониторе**, при этом монитор гарантирует, что в каждый момент времени процедура монитора может использовать только один процесс. Остальные процессы, заинтересованные в вызове процедуры монитора, ставятся в очередь к монитору.

Как правило, монитор оперирует переменными типа conditional (условие) с помощью двух функций wait и signal (это системные вызовы). wait блокирует процесс, signal - разблокировывает его.

Расмотрим три классических монитора: простой, кольцевой буфер, читатели-писатели.

Простой монитор обеспечивает выделение определенного ресурса произвольному числу процессов.

Код монитора (!!!):

```
resource: monitor;
```

```
var
        busy: logical;
        x: conditional;

procedure acquire;
        begin
                if busy then wait(x);
                busy := true;
        end;

procedure release;
        begin
                busy := false;
                signal(x);
        end;

begin
        busy = false;
end;
```

Когда к монитору обращается процесс для захвата ресурса, они вызывает функцию acquire. Если значение логической переменной busy - истина, то по переменной x выполняется системный вызов wait. В результате значение логической переменной не меняется.

Если busy - ложь, то процесс, обратившийся к монитору с помощью acquire, получает доступ к ресурсу и продолжается без задержки. В результате, busy устанавливается значением истины.

Если процесс, который занимает ресурс, желает его освободить, то он вызывает процедуру монитора release и меняет значение busy на ложь, после этого вызывается функция signal, которая разблокирует другой процесс, который назодится в очереди к переменной типа собитие.

Для каждой причины, по которой монитор переводится в режим ожидания (???), назначается своя переменная типа условие. Какждая переменная типа условие - это способ обозначения соответствующей очереди.