Монитор Хоара

Хоар Чарлз Энтони Ричард 1934

```
monitor <имя_монитора>;
    <секция_описания>
    <секция_инициализации>
```

- Обеспечивается взаимное исключение доступа к ресурсам, описанным в мониторе
- С монитором связывается «механизм сигналов»

```
<секция_описания>::=
    <описание типов, констант,
     переменных, процедур, функций>
<секция_инициализации>::=
    begin
   <инициализация переменных>
    end <имя_монитора>.
```

Процедура доступа

Поставщик:

```
loop
   D:= Производство();
   Буфер.Записать(D);
endloop.
```

Потребитель:

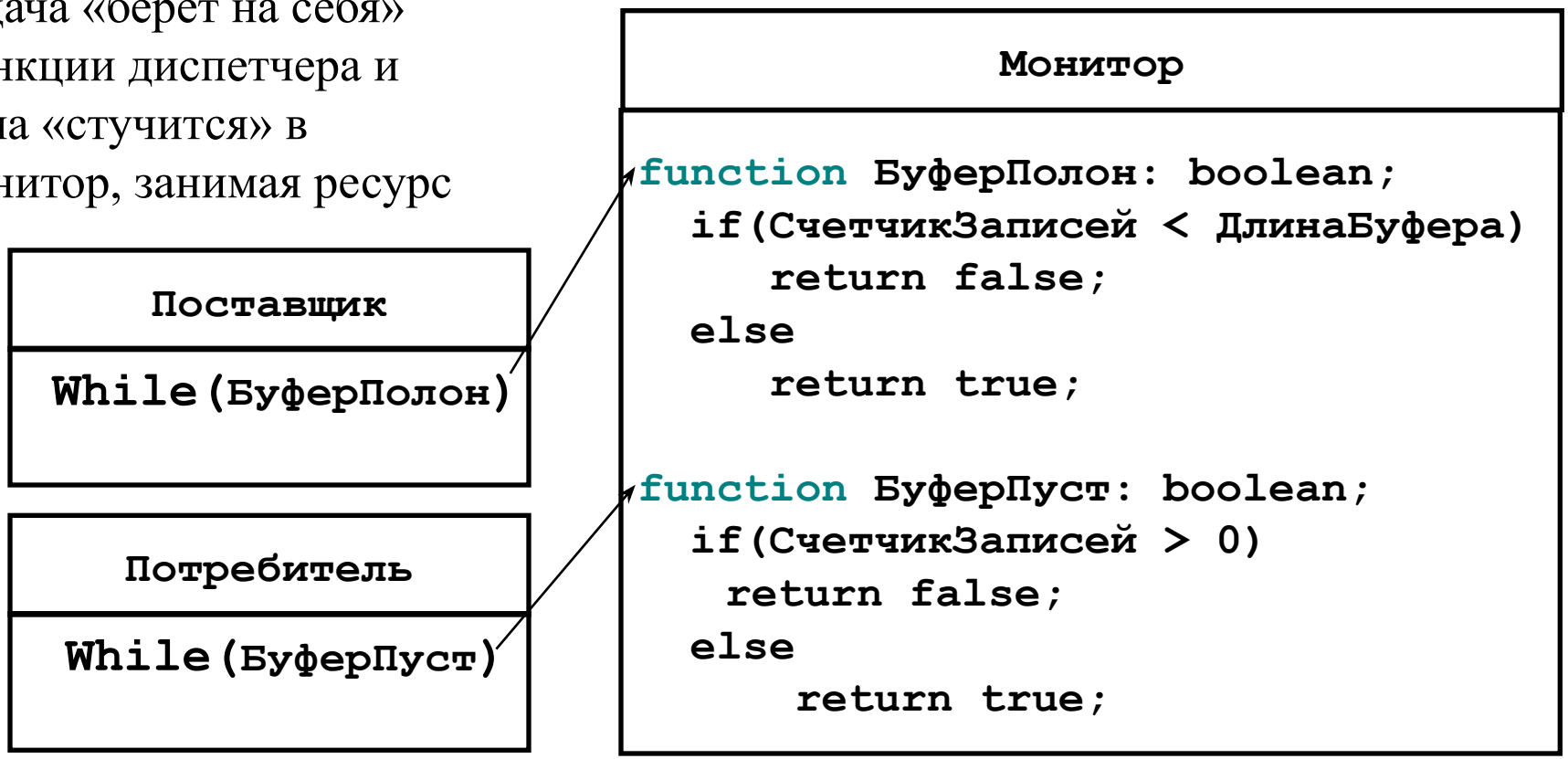
```
loop
   D:= Буфер.Прочитать();
   Обработка(D);
endloop.
```

```
monitor Буфер;
var СамБуфер:
array[1..ДлинаБуфера] of Данное;
var СчетчикЗаписей: integer;
procedure Записать(d: Данное);
      begin
      ...
      end;
function Прочитать(): Данное;
      begin
      ...
      end;
begin
   СчетчикЗаписей:= 0;
end Буфер.
```

Механизм сигналов

Желательна возможность «нотификации» изменений состояния ресурсов монитора. Пример: «Поставщик потребитель» - требуется оценивать состояние буфера (полон/пуст).

Решение 1 Плохое.
Задача «берет на себя» функции диспетчера и сама «стучится» в монитор, занимая ресурс



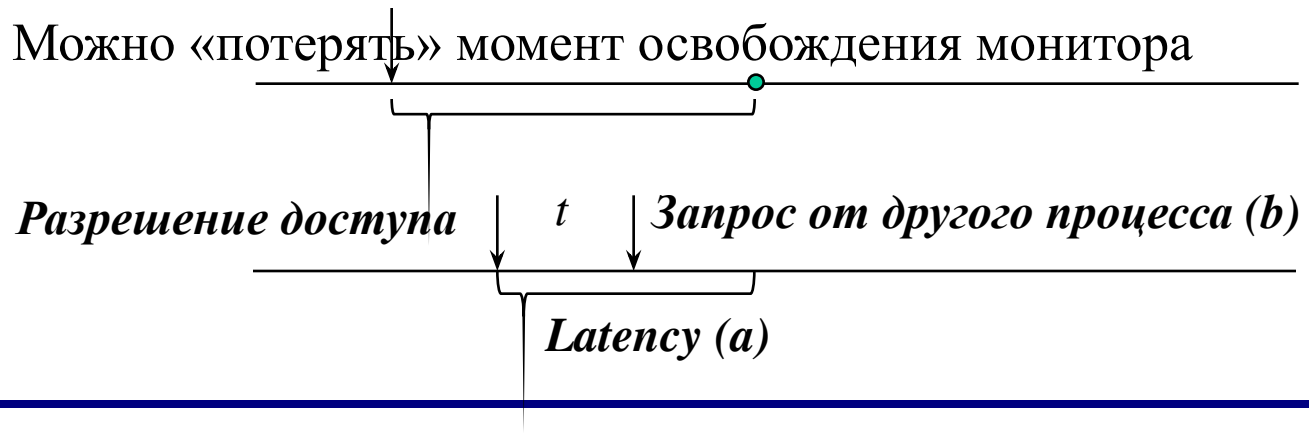
Решение 2 (совсем плохое)

Процесс «анализирует» возможность доступа и «засыпает» на некоторое время (ОС предоставляет возможность «заснуть» на время `t - sleep(t),` с передачей управления):

```
while(Условие){
    sleep(t);
}
```

a - Увеличивается latency

b - Можно «потерять» момент освобождения монитора



Сигналы - определение

Тип данных: **`type Сигнал`**

Операции над сигналами:

`Сигнал S = new Сигнал();` (Здесь строится очередь задач, ждущих получения сигнала S)

`wait(S):`

поставить активный процесс в очередь, связанную с сигналом S

`notify(S):`

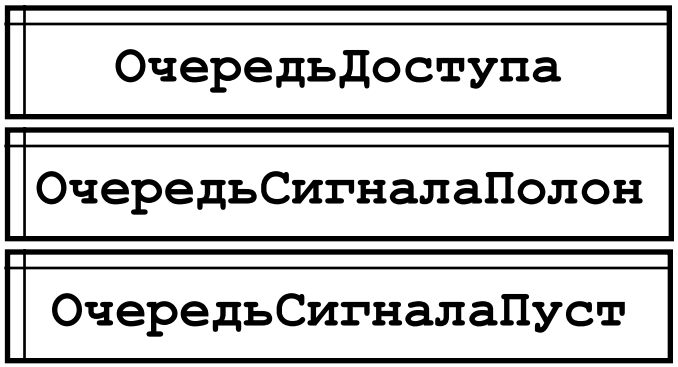
первый процесс из очереди S ставится в «очередь готовых»;

`check(S):Integer:`

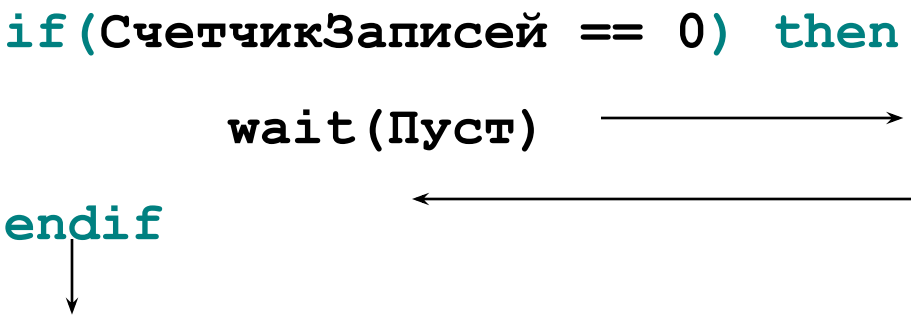
возвращает кол-во процессов, ждущих в очереди S

Пример – ресурсы монитора «Буфер»

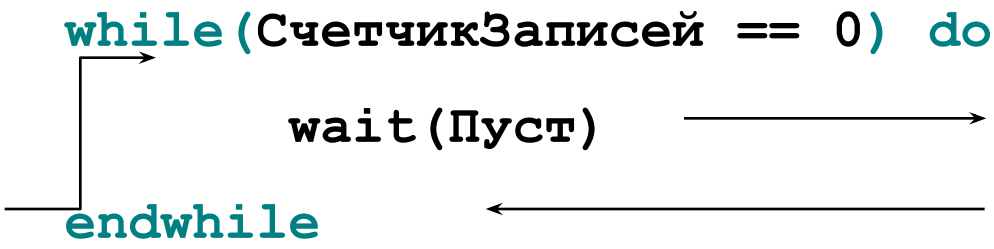
```
var Полон, Пуст: Сигнал;
procedure Записать(d: Данное);
  begin
    if(СчетчикЗаписей => ДлинаБуфера) then
       wait(Полон)
    endif
    ЗаписатьДанноеВБуфер(d);
    СчетчикЗаписей:=  СчетчикЗаписей + 1;
    notify (Пуст)
end;
function Прочитать(): Данное
  begin
    if(СчетчикЗаписей = 0) then
       wait(Пуст)
    endif
    Прочитать:= ЧтениеЗаписиИзБуфера();
    СчетчикЗаписей:=  СчетчикЗаписей - 1;
    notify (Полон)
end;
```



Еще одна «тонкость» ...



Но до выполнения **wait(Пуст)** условие **СчетчикЗаписей == 0** может поменяться несколько раз («*На время выполнения процессов не накладывается никаких ограничений*»)



Так что многое, представленное ранее – не совсем корректно

Монитор в Java

```
class Buffer {
    int[] buf;
    public synchronized void put (int s) {
        if (counter < CAPACITY) {
            counter ++; write_to_buf;
            this.notify();
        } else {
            this.wait();
        };
    };
    public synchronized int get () {
        if (counter > 0) {
            counter--; read_from_buf;
            this.notify();
        } else {
            this.wait();
        };
        return buf[counter];
    }
}
```

Пример - задача «Читатели-Писатели»

- **M Читателей и N Писателей получают доступ к Информационному Фонду**
- **Реализовать механизм, позволяющий обеспечить следующее условие:**

  в каждый момент времени могут работать не более одного Писателя или не более M Читателей

Схема реализации

```
Читатель:
loop
    ЧП.НачалоЧтения();
    РаботаСФондомЧ();
    ЧП.КонецЧтения();
    РазноеЧ;
 endloop.
```

```
Писатель:
loop
    ЧП.НачалоЗаписи();
    РаботаСФондомП();
    ЧП.КонецЗаписи();
    РазноеП;
 endloop.
```

```
monitor ЧП;
var МожноЧитать,
    МожноПисать: Сигнал;
    КтоТоПишет: boolean;
    Читатели: 0..M;

procedure НачалоЧтения;
procedure КонецЧтения;
procedure НачалоЗаписи;
procedure КонецЗаписи;

begin
   КтоТоПишет:= false;
   Читатели:= 0;
end ЧП.
```

Начало и окончание чтения

```
procedure НачалоЧтения();
 begin
  while (КтоТоПишет)or(check(МожноПисать)>0)
        wait(МожноЧитать);
   Читатели:= Читатели + 1;
   notify (МожноЧитать)
 end;
```

```
procedure КонецЧтения();
  begin
     Читатели:= Читатели - 1;
     if(Читатели = 0) then
        notify (МожноПисать)
     endif
  end;
```

Начало и окончание записи

```
procedure НачалоЗаписи();
  begin
    while(Читатели > 0)or(КтоТоПишет)
       wait(МожноПисать);
    КтоТоПишет:= true;
  end;
```

```
procedure КонецЗаписи();
  begin
    КтоТоПишет:= false;
    if(check(МожноЧитать) > 0) then
       notify(МожноЧитать)
    else
       notify (МожноПисать)
    endif;
end;
```

Пример – монитор и семафоры

```
monitor Семафор;
var Счетчик: 0..1;
    S: Сигнал;
procedure P;

begin
  while(Счетчик = 0) do
    wait(S)
  endwhile;
  Счетчик = 0;
end;
procedure V;
begin
  Счетчик:= 1;
  notify (S);
end;

begin
  Счетчик:= 1;
end Семафор.
```

Задача Ri:

```
loop
      . . .
   Семафор.P();
   КритическаяСекция_i
   Семафор.V();
      . . .
 endloop.
```

```
Parbegin R1; R2;. . . Rn Parend.
```