Programação Concorrente

Monitores

Prof. Eduardo Alchieri

Monitores

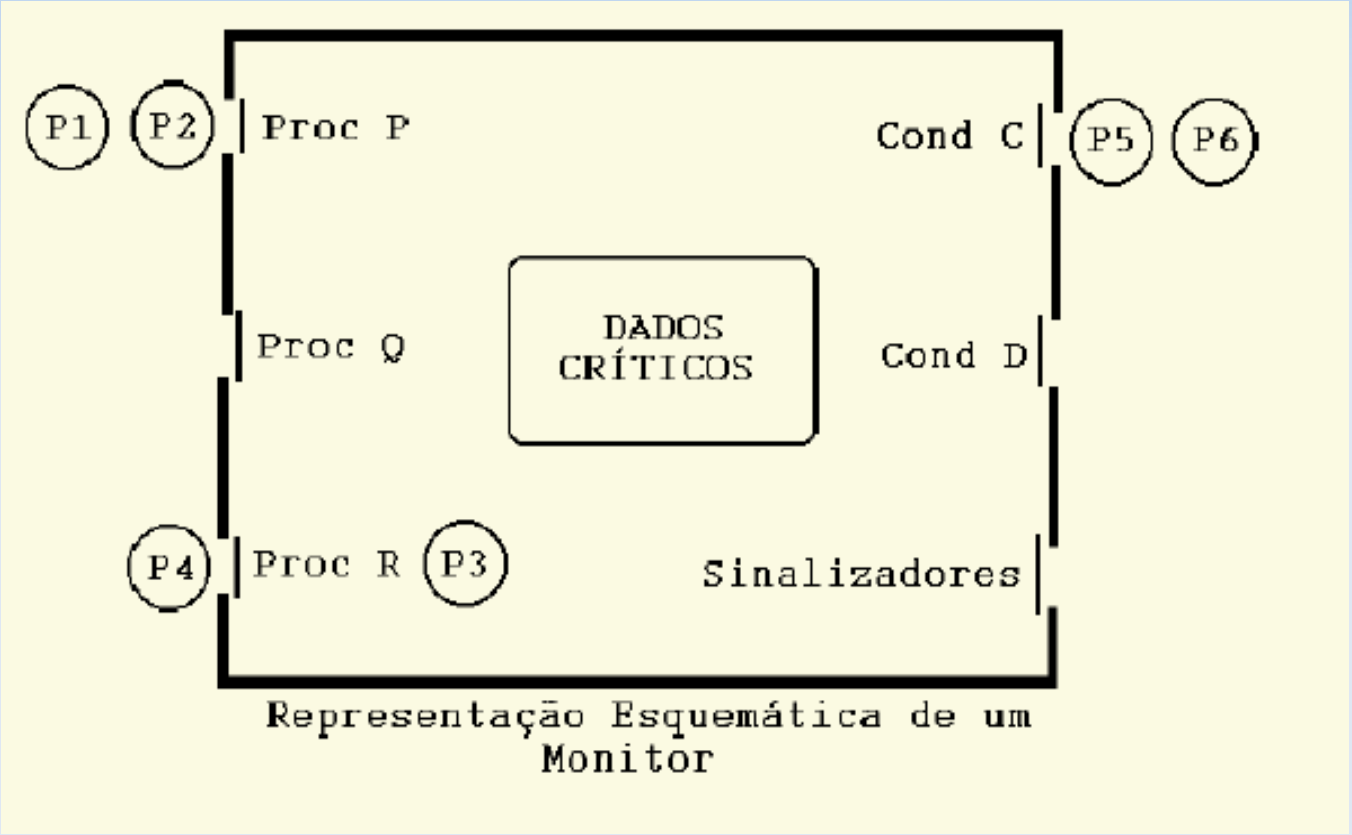
- **Monitores**
  - Um monitor é um conjunto de procedimentos, variáveis e estruturas de dados, todas agrupadas em um módulo especial
  - Monitores são mecanismos de sincronização de alto nível, que tentam tornar mais fácil o desenvolvimento de programas concorrentes
  - Sua característica mais importante é a implementação automática da exclusão mútua entre seus procedimentos, ou seja, somente um processo pode estar ativo dentro do monitor em um dado instante de tempo
  - Toda vez que algum processo chama um desses procedimentos, o monitor verifica se já existe outro processo executando algum procedimento do monitor
  - Caso exista, o processo ficará aguardando sua vez até que tenha permissão para executar

Monitores

- **Monitores**
  - Toda a implementação da exclusão mútua nos monitores é realizada pelo compilador e não mais pelo programador, como no caso do uso dos semáforos
    - Menos erros de programação
  - E como sincronizar processos?
    - Wait e Signal sobre condições definidas dentro do monitor
    - Uma thread que invoca wait fica suspensa até que outra invoque signal

Monitores

- Monitores



Representação Esquemática de um Monitor

Monitores

- Produtor\Consumidor usando monitores

```
procedure producer;
begin
 while true do
 begin
  produce_item;
  ProduceConsumer.enter;
 end;
end;
```

```
procedure consumer;
begin
 while true do
 begin
  ProduceConsumer.remove;
  consume_item;
 end;
end;
```

```
monitor ProducerConsumer
  condition full, empty;
  integer count;
      procedure enter;
      begin
        if count = N then wait(full);
        enter_item;
        count := count + 1;
        if count = 1 then signal(empty);
      end;
      procedure remove;
      begin
        if count = 0 then wait(empty);
        remove_item;
        count := count - 1;
        if count = N - 1 then signal(full);
      end;
      count := 0;
end monitor;
```

Monitores

- **Monitores**
  - **Java** é um exemplo de linguagem que permite o uso de monitor
    - Java suporta *threads* de usuário e também permite que métodos sejam agrupados em classes
    - Adicionando-se a palavra-chave ***synchronized*** à declaração de um método, Java garante que, uma vez iniciado qualquer thread executando esse método, a nenhuma outra thread será permitida executar qualquer outro método ***synchronized*** naquela classe
      - Wait
      - Notify (signal)
      - NotifyAll (broadcast)

Monitores

- Exercício: Como implementar um semáforo usando monitores?