

INE 5410

PROGRAMAÇÃO CONCORRENTE

Prof. Vitorio Bruno Mazzola
mazzola@inf.ufsc.br



Capítulo 3 MONITORES



Sincronização

- Sincronização envolvendo semáforos
 - Semáforo é um mecanismo de extrema importância para o controle de aplicações concorrentes, conforme apresentado
 - Apesar de eficiente, o sucesso de sua aplicação depende fortemente da habilidade do programador, o que é preocupante no caso de aplicações de grande porte;
 - Qualquer distração pode conduzir a resultados indesejáveis, incluindo travamentos (deadlocks) no sistema.

Monitores

- Origem
 - Idéias propostas por Hoare, Dijkstra e Brinch Hansen (1971) e implementada pelo último em sua linguagem concorrente – Concurrent Pascal



Edsger Dijkstra



Charles Antony Hoare



Per Brinch Hansen

Monitores

• Características

- Monitor tem sua operação baseada no funcionamento de semáforos;
- Constitui um mecanismo de mais fácil uso, porque o programador não precisa preocupar-se sobre onde localizar as operações de bloqueio ou liberação do acesso a um recurso;
- Programação é realizada num nível mais alto de abstração

Monitores

• Operação

- Dois processos num mesmo monitor não podem estar ativos simultaneamente;
- A exclusão mútua é assegurada pelo próprio mecanismo, ficando o programador liberado desta preocupação;
- Se um processo está ativo num mesmo monitor, outros processos que tentem acesso serão colocados numa fila de espera comum ao monitor.

Monitores

• Operação

- Entradas (procedimentos do monitor) acessam apenas variáveis locais do monitor... São impedidas de acessar variáveis declaradas fora do mesmo;
- De modo similar, variáveis locais de um monitor não podem ser acessadas por processos diretamente... Apenas pelas

Monitores

• Operação

- Duas operações, *wait* e *signal*, são utilizadas para garantir e bloquear o acesso, respectivamente;
- Uma operação *wait* vai suspender um processo associado a uma variável do tipo *condição*;
- O efeito disso é a liberação do monitor para aceitar um novo processo em sua entrada

Monitores

- Operação

- Executando uma operação signal sobre uma variável condição, o processo deixa o monitor;
- O processo que estiver ocupando a primeira posição da fila de espera da variável considerada será, então, sinalizado;
- O processo sinalizado retoma então sua execução, a partir do ponto imediatamente posterior à execução da sua operação wait.

Monitores

- Operação

- Caso não existam processos na fila relacionada à variável sobre a qual a operação signal executou, o resultado desta não provoca efeito algum;
- O programador pode definir tantas variáveis de condição quantas sejam as razões consideradas para que processos aguardem;

Monitores

- Comparando com semáforos

- No caso de variáveis semáforos, todo o processo deve ter as operações P e V (ou wait e signal) do semáforo convenientemente localizadas para garantir a exclusão mútua e controlar corretamente a concorrência;
- Isto já não acontece no caso dos monitores... A única forma de um processo ter acesso a recursos gerenciados pelo monitor é tendo acesso a ele através de suas entradas

Monitores

- Implementação no Pascal FC

- No Pascal-FC, a semântica do monitor é inspirada na semântica proposta por Hoare, segundo a apresentação anterior;
- Variáveis Condição sobre as quais se pode realizar três operações distintas:
 - delay(cond);
 - resume(cond);
 - empty(cond);

Monitores

• Implementação no Pascal FC

- **delay(cond)**
 - bloqueia um processo, liberando o monitor e colocando o processo que a executou numa fila de espera associada à variável *cond*;
- **resume(cond)**
 - libera um processo na fila associada à variável... Se não existirem processos à espera, a operação não tem nenhum efeito;
- **empty(cond)**
 - função booleana que retorna *true* se a fila de processos associada à variável estiver vazia.

Monitores

• Exemplo de aplicação

```
PROGRAM PROCS;  
  
MONITOR Out;  
EXPORT print;  
  
PROCEDURE print(VAR procid : INTEGER);  
BEGIN  
    WRITELN('Processo ',procid:3);  
END; (* print *)  
  
BEGIN  
    END; (* Out *)
```

Monitores

• Exemplo de aplicação

```
PROCESS Type ProcType (pid : INTEGER);  
  
BEGIN  
    REPEAT  
        SLEEP(4);  
        Out.print(pid);  
    FOREVER  
END; (* PROCTYPE *)  
  
VAR  
    P : ARRAY[1..12] OF ProcType;  
    I : INTEGER;
```

Monitores

• Exemplo de aplicação

```
BEGIN (* main *)  
    COBEGIN  
        FOR I:= 1 TO 12  
            DO P[I](I);  
        COEND;  
    END.
```

Monitores

- Exemplo de aplicação

```
PROGRAM INCREMENTO;  
VAR Conta : INTEGER;  
  
MONITOR Contagem;  
EXPORT inc, prn;  
  
PROCEDURE inc (VAR n : INTEGER);  
BEGIN  
    n := n + 1;  
END; (* inc *)
```

Monitores

- Exemplo de aplicação

```
PROCEDURE prn (VAR n : INTEGER);  
BEGIN  
    WRITELN(n:3);  
END; (* prn *)  
  
BEGIN  
END; (* Contagem *)
```

Monitores

- Exemplo de aplicação

```
PROCESS Type ProcType (pid : INTEGER);  
VAR I : INTEGER;  
BEGIN  
    FOR I := 1 TO 25  
    DO  
        BEGIN  
            Contagem.inc(Conta);  
            Contagem.prn(pid);  
        END;  
    END; (* PROCTYPE *)  
  
VAR  
    P : ARRAY[1..8] OF ProcType;  
    I : INTEGER;
```

Monitores

- Exemplo de aplicação

```
BEGIN (* main *)  
    COBEGIN  
        FOR I:= 1 TO 8  
            DO P[I](I);  
        COEND;  
    WRITELN('Contagem Total +++> ', Conta:3);  
END.
```

Monitores

• Exemplo de aplicação

```
PROGRAM PRODCONS;  
MONITOR Buffer;  
EXPORT put, take;  
  
CONST  
  BufMax = 4;  
  
VAR  
  store: ARRAY[1..BufMax] OF CHAR;  
  count, nextin, nextout: integer;  
  notfull, notempty: condition;
```

Monitores

• Exemplo de aplicação

```
PROCEDURE put(VAR ch: CHAR);  
BEGIN  
  if count > BufMax then  
    delay(notfull);  
  store[nextin] := ch;  
  count := count + 1;  
  nextin := (nextin + 1) MOD (BufMax + 1);  
  resume(notempty)  
END; (* put *)
```

Monitores

• Exemplo de aplicação

```
PROCEDURE take(VAR ch: char);  
BEGIN  
  IF COUNT = 0  
  THEN delay(notempty);  
  ch := store[nextout];  
  count := count - 1;  
  nextout := (nextout + 1) MOD (BufMax + 1);  
  resume(notfull)  
end; (* take *)
```

Monitores

• Exemplo de aplicação

```
(* body of BUFFER *)  
BEGIN  
  count := 0;  
  nextin := 0;  
  nextout := 0;  
END; (* BUFFER *)  
  
PROCESS producer;  
VAR  
  local: char;  
begin  
  FOR local := 'a' to 'z'  
  DO Buffer.put(local);  
end; (* PRODUCER *)
```

Monitores

- Exemplo de aplicação

```
PROCESS Consumer;  
VAR ch : char;  
BEGIN  
  REPEAT  
    BUFFER.TAKE(ch);  
    WRITE(ch)  
  UNTIL ch = 'z';  
  WRITELN;  
END; (* Consumer *)  
  
BEGIN (* main *)  
  COBEGIN  
    Producer;  
    Consumer  
  COEND;  
END.
```

Monitores

- Modos de Sinalização

- Quando um processo realiza uma liberação numa variável condição e a fila de processos desta variável não está vazia, é possível prever duas possibilidades;
- Estas duas possibilidades de sinalização são conhecidas pelos termos:
 - Signal and continue
 - Signal and wait

Monitores

- Modos de Sinalização

- Signal and continue
 - o processo que executou a operação pode continuar sua execução;
 - é um processo não preemptivo, ou seja, o processo sinalizado retorna ao grupo de processos aguardando a posse do monitor poderá, eventualmente, voltar a executar após o término do processo que sinalizou

Monitores

- Modos de Sinalização

- Signal and wait
 - o processo que executou a operação é imediatamente interrompido pelo processo liberado;
 - o processo interrompido retorna ao grupo de processos aguardando a posse do monitor
 - o processo liberado assume o controle do monitor.

Monitores

- Sinalização no Pascal-FC

- Variante do Signal and wait

- o processo que executou a operação é imediatamente interrompido pelo processo liberado;
 - o processo interrompido liberado e vai para uma fila específica (*fila do sinalizador*) aguardando a posse do monitor;
 - o processo liberado assume o controle do monitor.

Monitores

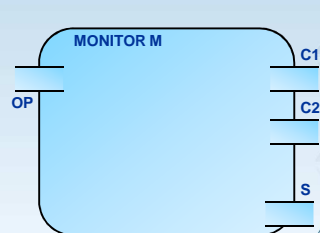
- Sinalização no Pascal-FC

- Variante do Signal and wait

- quando um processo deixa o monitor ou vai para a fila de espera de alguma variável condição (não tendo liberado nenhum processo em fila de qualquer variável condição), a ordem de liberação dos processos é ...
 - processos na *fila do sinalizador*;
 - processos em filas de operação do monitor.

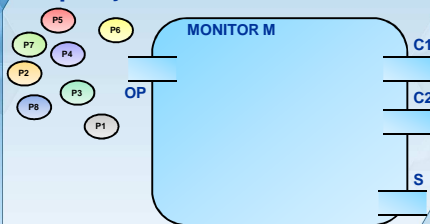
Monitores

- Exemplo de aplicação



Monitores

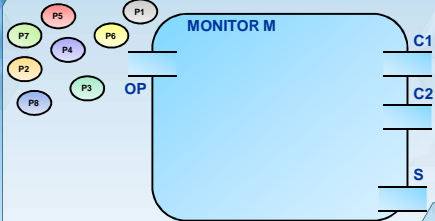
- Operação Monitor – Cenário 1



P4 finaliza execução de M.op (deixa o monitor)

Monitores

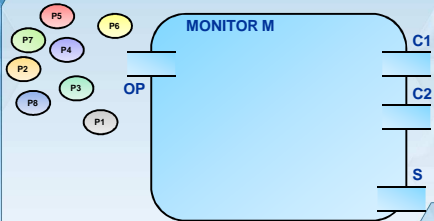
• Operação Monitor – Cenário 2



P2 finaliza M.op (deixa o monitor)

Monitores

• Operação Monitor – Cenário 2



P4 finaliza M.op (deixa o monitor)

Monitores

• Exemplo de Aplicação

```
PROGRAM piscina; (*versao monitor *)
VAR st : ARRAY[1..14] OF char;

MONITOR cesto;
EXPORT pega, larga;
VAR ncesto : integer;
    ccesto : condition;

PROCEDURE pega;
BEGIN
    IF ( ncesto = 0 ) THEN delay ( ccesto );
    ncesto := ncesto - 1;
END;
```

Monitores

• Exemplo de Aplicação

```
PROCEDURE larga;
BEGIN
    ncesto := ncesto + 1;
    resume ( ccesto );
END;

BEGIN
    ncesto := 5;
END;(*monitor cesto *)
```

Monitores

• Exemplo de Aplicação

```
MONITOR cabine;  
EXPORT pega, larga;  
VAR ncabine : integer;  
    ccabine : condition;  
  
PROCEDURE pega;  
BEGIN  
    IF( ncabine = 0 ) THEN delay ( ccabine );  
    ncabine := ncabine - 1;  
END;  
  
PROCEDURE larga;  
BEGIN  
    ncabine := ncabine + 1;  
    resume ( ccabine );  
END;
```

Monitores

• Exemplo de Aplicação

```
BEGIN  
    ncabine := 2;  
END;(*monitor cabine *)  
  
MONITOR interface;  
EXPORT alterast;  
VAR i : INTEGER;  
  
PROCEDURE alterast ( nid: integer; est : char );  
VAR i : integer;  
BEGIN  
    st[nid] := est;  
    FOR i := 1 TO 14 DO write ( ' ',st[i]);  
    writeln; writeln;  
END;
```

Monitores

• Exemplo de Aplicação

```
BEGIN  
END; (* interface *)  
  
PROCESS TYPE tpnadador ( nid : integer );  
BEGIN  
    REPEAT  
        interface.alterast( nid, 'A');  
        sleep( random ( 10 ) + 2 );  
        cesto.pega;  
        cabine.pega;  
        interface.alterast ( nid,'D');  
        sleep( random ( 8 ) + 2 );  
        cabine.larga;  
        interface.alterast ( nid,'N');
```

Monitores

• Exemplo de Aplicação

```
        cabine.larga;  
        interface.alterast ( nid,'N');  
        sleep( random ( 10 ) + 5 );  
        cabine.pega;  
        interface.alterast ( nid,'V');  
        sleep( random ( 8 ) + 2 );  
        cesto.larga;  
        cabine.larga  
    FOREVER;  
END; (* tpnadador *)
```

Monitores

- Exemplo de Aplicação

```
VAR nadador : array [ 1..14 ] of tpnadador;  
i: integer;  
BEGIN  
  COBEGIN  
    FOR i:= 1 TO 14 DO nadador [ i ] ( i );  
  COEND;  
END.
```

Monitores

- Dualidade entre Monitores e Semáforos

```
MONITOR semaforo;  
EXPORT p,v;  
VAR valor : integer;  
    cond : condition;  
  
PROCEDURE p (pid : integer);  
BEGIN  
  IF valor = 0 THEN delay(cond);  
  valor := valor - 1;  
END;
```

Monitores

- Dualidade entre Monitores e Semáforos

```
PROCEDURE v (pid : integer);  
BEGIN  
  valor := valor + 1;  
  resume(cond);  
END;  
  
BEGIN  
  valor := 1;  
END;
```