Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Tema 4. Monitores

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Programación Concurrente y Distribuida

Septiembre 2021

1. Introducción

Permite un control estructurado de la exclusión mútua y un mecanismo de control de la sincronización igual de versátil que los semáforos.
Permite encapsular datos protegiendo el uso indebido de las variables usadas.
El acceso a las variables del monitor solo es posible mediante el conjunto de funciones exportadas por el monitor al exterior.

Los programas que usan monitores tienen dos tipos de procesos: pasivos (implementan el monitor y los datos solo pueden ser manipulados por ellos)y activos (realizan llamadas a los procedimientos exportados por el monitor).

Los procesos activos no se preocupan de la implementación, solo del uso de las operaciones.

2. Definición de Monitor

El monitor contiene las variables que representan el estado del recurso y los procedimientos que implementan las operaciones. Garantiza la exclusión mutua de los procedimientos que contiene. Se compone de los siguientes elementos:

- Conjunto de variables de estado "permanentes".
- Un código de la inicialización de las variables "permanentes".
- Conjunto de procedimientos internos.
- Declaración de procedimientos exportados.

El control de la exclusión mutua sobre el monitor se basa en una cola asociada al monitor. Cuando un proceso activo está ejecutando código monitor y otro proceso intenta ejecutar otro de los procedimientos del monitor, éste queda esperando en la cola.

Cuando un proceso activo abandona el monitor, el primero proceso esperando en la cola obtiene acceso al procedimiento que provocó el bloqueo.

Si un proceso activo abandona el monitor y la cola está vacía, el monitor queda libre para ser usado.

```
monitor incremento;
  var i: integer;
  export inc, leer;

  procedure inc;
  begin
    i:=i+1;
  end;

  procedure leer(var valor:integer);
  begin
    valor:=i;
  end;

begin
  i:=0;
end;
```

3. Condición de sincronización en monitores

La exclusión mutua está garantizada, pero se necesita un mecanismo para detener la ejecución de un proceso cuando no se cumpla una condición. Las variables condición representan colas FIFO que están ini-

cialmente vacías.

Las operaciones que permiten que los procesos se bloqueen o salgan de estas colas son delay y resume.

- Delay (c): Siendo c una variable condition hace que el proceso que la ejecute se bloquee al final de la cola asociada a la variable. Antes de bloquearse, el proceso libera la exclusión mutua del monitor para que otros procesos puedan acceder a él.
- Resume (c): Desbloquea el primer proceso de la cola asociada a c. El proceso que realiza la operación resume abandona el monitor, que es tomado por el proceso desbloqueado. Una vez finalice el proceso desbloqueado, el proceso que realizó el resume tiene preferencia.

- empty(c): Devuelve un booleano que representa si la cola asociada a c está vacía o no.

Implementación de un semáforo binario mediante monitores:

```
monitor semaforo_binario;
  var sem: boolean;
      csem: condition;
  export wait, signal;

  procedure wait;
  begin
    if sem then delay(csem);
    sem := true;
  end;

  procedure signal;
  begin
    if not empty(csem) then resume(csem);
    else sem:=false;
  end;

begin
  sem:=false;
end;
```

4. Resolución de problemas usando monitores

4.1. Problema del productor-consumidor

```
process productor; {PRODUCTOR}
var
        local: char;
begin
        for local := 'a' to 'z' do Buffer.poner(local);
end;

process consumidor; {CONSUMIDOR}
var
        ch: char;
begin
        repeat
                Buffer.sacar(ch);
                write(ch,' ')
        until ch = 'z';
        writeln
end;
```

```
monitor Buffer;
        export
                poner, sacar;
        const
                MAX = 5;
        var
                recurso: array[0..4] of char;
                elementos: integer;
                nolleno, novacio: condition;
                cola, frente: integer;

        procedure poner(ch: char);
        begin
                if elementos = MAX then delay(nolleno);
                recurso[cola] := ch;
                elementos := elementos + 1;
                cola := (cola + 1) mod MAX;
                resume(novacio)
        end;

        procedure sacar(var ch: char);
        begin
                if elementos = 0 then delay(novacio);
                ch := recurso[frente];
                elementos := elementos - 1;
                frente := (frente + 1) mod MAX;
                resume(nolleno)
        end;

        begin
                elementos := 0;
                frente := 0;
                cola := 0
        end;
```

4.2. Problema de los lectores-escritores

```
process type Lector(id:integer);
begin
        repeat
                le.entraleer;
                writeln('Lector ',id,' leyendo');
                sleep(random(3));
                le.saleleer;
                writeln('Lector ',id,' sale de leer');
        forever
end;

process type Escritor(id:integer);
begin
        repeat
                le.entraescribir;
                writeln('Escritor ',id,' leyendo');
                sleep(random(3));
                le.saleescribir;
                writeln('Escritor ',id,' sale de escribir');
        forever
end;
```

```
monitor le;

        export
                entraleer, entraescribir, saleleer, saleescribir;

        var
                nl: integer;
                escribiendo: boolean;
                lector, escritor: condition;

        procedure entraleer;
        begin
                if escribiendo or not empty(escritor) then delay(lector);
                nl:=nl+1;
                resume(lector);
        end;

        procedure saleleer;
        begin
                nl:=nl-1;
                if nl = 0 then resume(escritor);
        end;

        procedure entraescribir;
        begin
                if (nl <> 0) or escribiendo then delay(escritor);
                escribiendo:=true;
        end;

        procedure saleescribir;
        begin
                escribiendo:=false;
                if not empty(lector) then resume(lector)
                else resume(escritor);
        end;

        begin
                nl:=0;
                escribiendo:=false;
        end;
```

4.3. Problema de la comida de los filósofos

```
process type Filosofo(n: integer);
begin
        repeat
                sleep(random(5));    (* THINKING *)
                Tenedores.coger(n);
                sleep(random(5));    (* EATING *)
                Tenedores.soltar(n)
        forever
end;
```

```
monitor Tenedores;
        export
                coger, soltar;
        var
                tenedores: array [0..4] of integer;
                espera: array [0..4] of condition;
                i: integer;

        procedure coger(i: integer);
        begin
                if tenedores[i] <> 2 then
                begin
                        writeln('Filosofo ',i,' esperando');
                        delay(espera[i])
                end;
                tenedores[(i+1) mod 5] := tenedores[(i+1) mod 5] - 1;
                tenedores[(i+4) mod 5] := tenedores[(i+4) mod 5] - 1;
                writeln('Filosofo ',i,' come')
        end;

        procedure soltar(i: integer);
        begin
                writeln('Filosofo ',i,' termina');
                tenedores[(i+1) mod 5] := tenedores[(i+1) mod 5] + 1;
                tenedores[(i+4) mod 5] := tenedores[(i+4) mod 5] + 1;
                if tenedores[(i+1) mod 5] = 2 then resume(espera[(i+1) mod 5]);
                if tenedores[(i+4) mod 5] =2 then resume(espera[(i+4) mod 5])
        end;

        begin
                for i := 0 to 4 do tenedores[i] := 2
        end;
```

5. Semántica de la operación resume

Dado que la operación resume despierta a uno de los procesos que esperan en el monitor, la implementación debe decidir qué hacer tras esta operación:

1. **Desbloquear y continuar**: El proceso desbloqueado no comienza a ejecutarse hasta que el que realiza el resume no sale del monitor. El proceso que realiza el resume puede modificar alguna variable de estado del monitor, por lo que el desbloqueado debe comprobar la condición tras ser desbloqueado.

2. **Retorno forzado**: Obliga al proceso que realiza el resume finalice, de forma que dicho proceso sale del monitor cediéndolo al proceso desbloqueado. El proceso desbloqueado no debe volver a comprobar la condición.

3. **Desbloquear y esperar**: El proceso que realiza el resume cede el monitor al proceso desbloqueado. El proceso que hace el resume deberá volver a competir por la exclusión mutua del monitor. El proceso desbloqueado no tiene que evaluar de nuevo la condición, pero se perjudica al proceso desbloqueador.

4. **Desbloquear y espera urgente**: Asocia al monitor una nueva cola de cortesía. El proceso desbloqueador espera en la cola de cortesía que tiene preferencia para la entrada al monitor.