Università degli Studi di Padova

Risorse protette

SCD

Anno accademico 2003/4
Corso di Sistemi Concorrenti e Distribuiti

Tullio Vardanega, tullio.vardanega@math.unipd.it

Limiti del modello base

- **È desiderabile ottenere mutua esclusione e sincronizzazione condizionale senza dover impiegare un processo servente**
- **È anche desiderabile che la corrispondente astrazione abbia potenza espressiva non inferiore al modello del processo servente**
- **I modelli di riferimento sono il *monitor* di Hoare e le regioni critiche condizionali**
  - **1972, Per Brinch-Hansen, "**Structured Multiprogramming**", CACM vol. 15(7), pp. 574-578**

Mutua esclusione - 1

- **Requisiti**
  1. **Accesso in mutua esclusione in scrittura**
  2. **Accesso potenzialmente concorrente in sola lettura**
  3. **Distinzione strutturale tra operazioni di sola lettura ed operazioni R/W**

```
protected type Shared_Integer (Initial_Value : Integer) is
  function Read return Integer;
  procedure Write (Value : Integer);
private
  The_Integer : Integer := Initial_Value;
end Shared_Integer;
```

```
protected body Shared_Integer is
  function Read return Integer is
  begin
    return The_Integer;
  end Read;
  procedure Write (Value : Integer) is
  begin
    The_Integer := Value;
  end Write;
end Shared_Integer;
```

Letture tra loro concorrenti

Scritture mutuamente esclusive tra loro e con ogni lettura

Mutua esclusione - 2

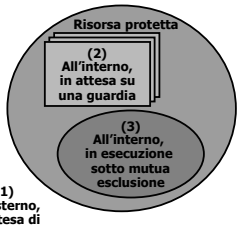
- **Vantaggi rispetto all'impiego di un processo servente**
  - **Risparmio di risorse a tempo d'esecuzione**
    - Un'entità passiva, con un agente implicito di protezione, contro un'entità attiva programmata come agente di mutua esclusione
  - **Minore complessità di terminazione**
    - Un'entità passiva non termina, semplicemente viene rimossa non appena il suo ambito (*scope*) cessa di esistere
    - L'onere è tutto e solo sui processi cliente

Sincronizzazione condizionale - 1

- **Requisiti**
  - **Possibilità per un processo cliente di sincronizzarsi con una condizione (ed un servizio ad essa associato) invece che con un altro processo**
  - **Possibilità per un processo cliente di sospendersi in attesa che la condizione attesa si verifichi**
- **Possibile soluzione**
  - **Punto di accesso (*entry*) offerto da una risorsa protetta, sul quale applica una guardia Booleana**
    - La guardia opera come **precondizione**
  - **L'accodamento su guardia chiusa avviene all'interno della risorsa protetta e non fuori di essa**
    - Così che l'attesa sulla guardia non comporti rischio di ***starvation***
    - Politica base di ordinamento in coda: FIFO

Sincronizzazione condizionale - 2

Il modello a "guscio d'uovo"



- **2 livelli di protezione**
  - **Esecuzione in mutua esclusione (3)**
  - **Attesa su guardia booleana (precondizione) senza rischi di *starvation***
- **La valutazione della guardia avviene in mutua esclusione**

Università degli Studi di Padova

Risorse protette

Esempio: un contenitore vincolato

```
Buffer_Size : constant Positive := 5;
type Index is mod Buffer_Size;
subtype Count is Natural range 0 .. Buffer_Size;
type Buffer_T is array(Index) of Any_Type;
protected type Bounded_Buffer is
   entry Get ( Item :    out Any_Type );
   entry Put ( Item : in     Any_Type );
private
   First : Index := Index'First; -- 0
   Last  : Index := Index'Last;  -- 4
   In_Buffer : Count := 0;
   Buffer : Buffer_T;
end Bounded_Buffer;
protected body Bounded_Buffer is
   entry Get ( Item :    out Any_Type ) when In_Buffer > 0 is
   begin -- first read then move pointer
      Item := Buffer(First);
      First := First + 1;
      In_Buffer := In_Buffer - 1;
   end Get;
   entry Put ( Item : in     Any_Type ) when In_Buffer < Buffer_Size is
   begin --  first move pointer then write
      Last := Last + 1;
      Buffer(Last) := Item;
      In_Buffer := In_Buffer + 1;
   end Put;
end Bounded_Buffer;
```

Parte pubblica

Parte privata

Specifica

Guardie

Università degli Studi di Padova

Risorse protette

Sincronizzazione condizionale - 3

- **Usando il costrutto `select` un processo cliente può limitare il tempo d'attesa su un punto d'accesso protetto**
- **Ad ogni istante, un punto d'accesso protetto può essere**
  - **Aperto, se la sua guardia, quando valutata, è vera**
  - **Chiuso, se la sua guardia, quando valutata, è falsa**
- **Una guardia di punto d'accesso protetto viene rivalutata**
  - **Ad ogni richiesta d'accesso la cui guardia abbia una componente che possa essere cambiata dall'ultima valutazione**
    - Non a seguito di una chiamata di una funzione protetta!
  - **Ad ogni completamento d'esecuzione R/W in mutua esclusione entro la risorsa protetta, ove vi siano processi accodati su una guardia le cui componenti potrebbero essere cambiate dall'ultima valutazione**

Università degli Studi di Padova

Risorse protette

Sincronizzazione condizionale - 4

- **Una risorsa protetta è detta "in uso per lettura" (*read lock*) quando ≥ 1 processi stiano eseguendo una sua funzione**
- **La risorsa è detta "in uso R/W" (*R/W lock*) quando un processo stia eseguendo una sua procedura o *entry***
  - **Un processo in possesso di una risorsa protetta può effettuare chiamate di altri servizi della stessa risorsa, i quali vengono immediatamente soddisfatti senza soggiacere a competizione**
  - **Ciò non può però avvenire indirettamente, ossia quando la chiamata viene effettuata da un sottoprogramma esterno alla risorsa ma chiamato dall'interno di essa → situazione erronea**
- **In entrambi i casi ogni altra chiamata viene trattenuta all'esterno della risorsa**

Università degli Studi di Padova

Risorse protette

Protocollo d'accesso - 1

1. Se la risorsa protetta (RP) è sotto "*read lock*" e la richiesta è di funzione, questa viene eseguita e si passa al punto **14**.
2. Se RP è sotto "*read lock*" e la richiesta è di procedura od *entry*, la chiamata è differita fin quando vi siano processi attivi all'interno di RP
3. Se RP è sotto "*R/W lock*", la chiamata viene differita fin quando vi siano processi attivi all'interno di RP i quali abbiano requisiti di accesso potenzialmente in conflitto con la chiamata
4. Se RP non è in uso e la chiamata è di funzione, RP assume un "*read lock*" e si passa al punto **5**.
5. La funzione di RP viene eseguita e si passa al punto **14**.
6. Se RP non è in uso e la chiamata è di procedura od *entry*, RP assume un "*R/W lock*" e si passa al punto **7**.
7. Se la chiamata è di procedura, questa viene eseguita e si passa al punto **10**., altrimenti si passa al punto **8**.

Università degli Studi di Padova

Risorse protette

Protocollo d'accesso - 2

8. Se la chiamata è di *entry*, la corrispondente guardia è valutata e, se aperta, si esegue il corpo dell'*entry* e si passa al punto **10**, altrimenti si passa al punto **9**.
9. Se la guardia valuta chiusa, la chiamata è posta nella coda associata alla guardia, da questo momento iniziando la valutazione delle clausole di selezione del chiamante, e si passa al punto **10**.
10. Viene rivalutata ciascuna guardia che abbia chiamate in attesa e la cui espressione contenga variabili che possano essere cambiate dall'ultima valutazione e poi si passa al punto **11**.
11. Tra le guardia che valutassero aperte se ne seleziona una eseguendo il corpo della corrispondente *entry* e poi si torna al punto **10**, altrimenti si passa al punto **12**.
12. Se nessuna guardia con chiamate in attesa è aperta si passa al punto **13**.
13. Tra le chiamate differite all'esterno di RP si selezionano o tutte quelle di funzione, che richiedono "*read lock*", oppure una tra quelle che richiedono "*R/W lock*" e si eseguono i passi **5** o **7** o **8**; se non vi fossero chiamate il protocollo d'accesso completa
14. Quando non vi fossero più processi attivi all'interno di RP si passa al punto **13**.

Università degli Studi di Padova

Risorse protette

Protocollo d'accesso - 3

- **L'aspetto più delicato del protocollo riguarda i punti 10-11**
- **Le guardie vengono rivalutate a seguito dell'esecuzione di procedure od *entry*, oltre che, naturalmente, al punto 8**
- **Per ognuna di quelle che divenute aperta si esegue il corpo dell'*entry* corrispondente, ma non occorre specificare quale processo se ne prenda carico effettivamente!**
  - **Potrebbe ugualmente essere il processo che ha emesso la chiamata oppure quello la cui azione abbia aperto la guardia**
- **Occorre anche notare come le clausole temporali di selezione poste dal chiamante siano valutate solo a partire da quando la chiamata venga effettivamente accodata**

Università degli Studi di Padova

Risorse protette

Controllo d'accodamento - 1

- I punti d'accesso (sia di processo che di risorsa protetta) hanno un attributo `'Count` come funzione che ritorna il numero di processi attualmente in coda per quell'*entry*
- Per questo anche l'accodamento di chiamata richiede "*write lock*" sulla risorsa
- L'uso dell'attributo nelle guardia ne comporta la rivalutazione ad ogni accodamento di chiamata

Università degli Studi di Padova

Risorse protette

Controllo d'accodamento - 2

Esempio: visita ad una mostra, con ingresso a gruppi di 5

```
protected Guardian is
 entry Let_In;
private
 Open : Boolean := False;
end Guardian;
protected body Guardian is
 entry Let_In when Let_In'Count = 5 or Open is
 begin
  if Let_In'Count = 0 then
   Open := False;
  else
   Open := True;
  end if;
 end Let_In;
end Guardian;
```

Il 5o chiamante troverà la guardia chiusa, ma il suo accodamento causerà la modifica di `'Count`, ciò comportando una nuova valutazione della guardia, ora divenuta aperta. La 1a chiamata accodata modificherà `Open` così che la guardia resti aperta fino a che sia stato rilasciata la 5a chiamata accodata, la quale chiuderà di nuovo la guardia

Vediamone l'esecuzione …

Università degli Studi di Padova

Risorse protette

Controllo d'accodamento - 3

- L'attributo `'Count` fornisce grande potenza espressiva
- È bene però tenere a mente che usarlo nell'espressione di una guardia può spesso causare ≥2 valutazioni della guardia senza soluzione di continuità
  - Alla chiamata del punto d'accesso
  - Al possibile accodamento della richiesta in presenza di guardia chiusa

Università degli Studi di Padova

Risorse protette

Restrizioni d'uso

- L'esecuzione entro una risorsa protetta (così come entro una sincronizzazione) dovrebbe essere il più breve possibile
- Per limitarne la durata, sono considerate erronee le chiamate potenzialmente bloccanti effettuate entro azioni protette
  - Clausola selettiva (`select`)
  - Clausola d'accettazione (`accept`)
  - Chiamata di `entry`
  - Chiamata di sospensione (`delay [until]`)
  - Creazione o attivazione di processo
  - Chiamata di sottoprogramma potenzialmente bloccante
- La rilevazione di una situazione erronea determina il completamento del programma con l'eccezione `Program_Error`
- In caso di mancata rilevazione il programma può entrare in stallo

Università degli Studi di Padova

Risorse protette

Elaborazione e finalizzazione

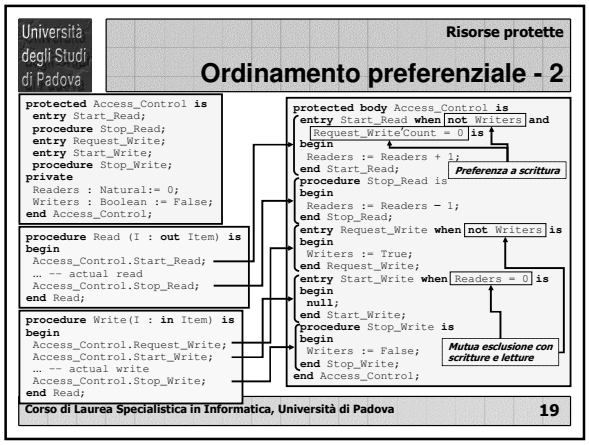
- L'elaborazione di una risorsa protetta avviene quando il suo ambito (*scope*) la richiede
- La sua finalizzazione però non può avvenire fin quando vi siano chiamate accodate sui suoi punti d'accesso
  - Ciò richiede che i corrispondenti processi vengano brutalmente completati con l'eccezione `Program_Error`

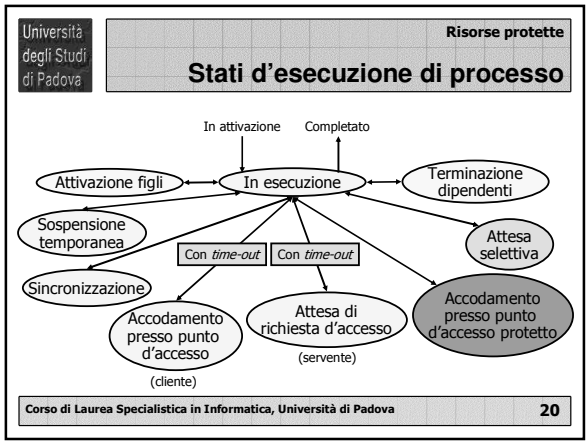
Università degli Studi di Padova

Risorse protette

Ordinamento preferenziale - 1

- Il vincolo di mutua esclusione può essere troppo rigido per problemi che richiedano ordinamento preferenziale dei servizi
  - Scritture, che richiedono "*write lock*" e necessitano di mutua esclusione, potrebbero essere preferibili a letture, che, se realizzate come funzione, richiederebbero solo "*read lock*" e potrebbero essere servite in parallelo
  - L'esecuzione di scritture e letture potrebbe essere bloccante e quindi non effettuabile entro risorsa protetta
- In questi casi la risorsa protetta aiuta a realizzare i necessari protocolli d'accesso ai servizi piuttosto che i servizi veri e propri


Università degli Studi di Padova
Risorse protette
Ordinamento preferenziale - 2
protected Access_Control is
entry Start_Read;
procedure Stop_Read;
entry Request_Write;
entry Start_Write;
procedure Stop_Write;
private
Readers : Natural:= 0;
Writers : Boolean := False;
end Access_Control;
procedure Read (I : out Item) is
begin
Access_Control.Start_Read;
… -- actual read
Access_Control.Stop_Read;
end Read;
procedure Write(I : in Item) is
begin
Access_Control.Request_Write;
Access_Control.Start_Write;
… -- actual write
Access_Control.Stop_Write;
end Read;
protected body Access_Control is
entry Start_Read when not Writers and Request_Write'Count = 0 is
begin
Readers := Readers + 1;
end Start_Read;
procedure Stop_Read is
begin
Readers := Readers - 1;
end Stop_Read;
entry Request_Write when not Writers is
begin
Writers := True;
end Request_Write;
entry Start_Write when Readers = 0 is
begin
null;
end Start_Write;
procedure Stop_Write is
begin
Writers := False;
end Stop_Write;
end Access_Control;
Preferenza a scrittura
Mutua esclusione con scritture e letture
Corso di Laurea Specialistica in Informatica, Università di Padova
19


Università degli Studi di Padova
Risorse protette
Stati d'esecuzione di processo
In attivazione
Completato
Attivazione figli
In esecuzione
Terminazione dipendenti
Sospensione temporanea
Attesa selettiva
Con time-out
Con time-out
Sincronizzazione
Accodamento presso punto d'accesso
(cliente)
Attesa di richiesta d'accesso
(servente)
Accodamento presso punto d'accesso protetto
Corso di Laurea Specialistica in Informatica, Università di Padova
20