

Università degli Studi di Padova

Risorse protette



Anno accademico 2004/5
 Corso di Sistemi Concorrenti e Distribuiti
 Tullio Vardanega, tullio.vardanega@math.unipd.it

Corso di Laurea Specialistica in Informatica, Università di Padova 1/21

Università degli Studi di Padova

Risorse protette

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

Corso di Laurea Specialistica in Informatica, Università di Padova 2/21

Università degli Studi di Padova

Risorse protette

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;
    
```

Lecture tra loro concorrenti

Scritture mutuamente esclusive tra loro e con ogni lettura

Corso di Laurea Specialistica in Informatica, Università di Padova 3/21

Università degli Studi di Padova

Risorse protette

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

Corso di Laurea Specialistica in Informatica, Università di Padova 4/21

Università degli Studi di Padova

Risorse protette

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'accadimento 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


Corso di Laurea Specialistica in Informatica, Università di Padova 5/21

Università degli Studi di Padova

Risorse protette

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**

Corso di Laurea Specialistica in Informatica, Università di Padova 6/21

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

Corso di Laurea Specialistica in Informatica, Università di Padova 7/21

Università degli Studi di Padova Risorse protette

Sincronizzazione condizionale - 3

- ❑ Usando il costrutto `select` un processo cliente può limitare il proprio tempo d'attesa su un punto d'accesso protetto
- ❑ 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 entro la RP, ove vi siano processi accodati su una guardia le cui componenti potrebbero essere cambiate dall'ultima valutazione

Corso di Laurea Specialistica in Informatica, Università di Padova 8/21

Università degli Studi di Padova Risorse protette

Sincronizzazione condizionale - 4

- ❑ Una RP è detta "in uso per lettura" (*read lock*) quando ≥ 1 processi stiano eseguendo una sua funzione
- ❑ La RP 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 RP può effettuare chiamate ad altri servizi della stessa risorsa, e queste vengono immediatamente soddisfatte, 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 RP

Corso di Laurea Specialistica in Informatica, Università di Padova 9/21

Università degli Studi di Padova Risorse protette

Protocollo d'accesso - 1

1. Se la risorsa protetta (RP) è sotto "read lock" e la chiamata è a funzione, questa viene eseguita e si passa al punto 14.
2. Se RP è sotto "read lock" e la chiamata è a procedura od *entry*, la chiamata viene 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 a procedura od *entry*, RP assume un "R/W lock" e si passa al punto 7.
7. Se la chiamata è a procedura, questa viene eseguita e si passa al punto 10., altrimenti si passa al punto 8.

Corso di Laurea Specialistica in Informatica, Università di Padova 10/21

Università degli Studi di Padova Risorse protette

Protocollo d'accesso - 2

8. Se la chiamata è ad *entry*, la corrispondente guardia è valutata e, se aperta, si esegue il corpo dell'*entry* e poi si passa al punto 10, altrimenti si passa al punto 9.
9. Poiché la guardia è chiusa, la chiamata è posta nella coda associata alla guardia, e da questo momento inizia 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 guardie che risultassero 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 a 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.

Corso di Laurea Specialistica in Informatica, Università di Padova 11/21

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 al punto di chiamata (8)
- ❑ Per ognuna di quelle che valutano aperte si esegue il corpo dell'*entry* corrispondente
 - Il protocollo non deve specificare quale processo si debba prendere carico dell'esecuzione!
 - Potrebbe essere il processo che ha emesso la chiamata oppure ugualmente quello la cui azione abbia aperto la guardia
- ❑ Le clausole temporali di selezione poste dal chiamante sono valutate solo a partire da quando la chiamata venga effettivamente accodata

Corso di Laurea Specialistica in Informatica, Università di Padova 12/21

Università degli Studi di Padova Risorse protette

Controllo d'accodamento - 1

- ❑ Ogni punto d'accesso (sia di *server* che di risorsa protetta) possiede un attributo `\Count`, che è una funzione che ritorna il numero di processi a quel momento in coda sull'*entry*
- ❑ Per questo motivo, anche l'accodamento di chiamata richiede "write lock" sulla risorsa!
- ❑ L'uso dell'attributo `\Count` nell'espressione Booleana della guardia ne comporta l'obbligo di rivalutazione ad ogni accodamento di chiamata

Corso di Laurea Specialistica in Informatica, Università di Padova 13/21

Università degli Studi di Padova Risorse protette

Controllo d'accodamento - 2

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

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

L'N-esimo 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 chiamata accodata modificherà `open` così che la guardia resti aperta fino a che sia stato rilasciata la 5a chiamata accodata, la quale chiederà di nuovo la guardia

Vediamone l'esecuzione ...

Corso di Laurea Specialistica in Informatica, Università di Padova 14/21

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

Corso di Laurea Specialistica in Informatica, Università di Padova 15/21

Università degli Studi di Padova Risorse protette

Restrizioni d'uso - 1

- ❑ L'esecuzione entro una RP dovrebbe essere il più breve possibile
 - Anche quelle all'interno di una sincronizzazione
- ❑ Per limitarne la durata, si considerano erronee le chiamate potenzialmente bloccanti effettuate entro azioni protette

Corso di Laurea Specialistica in Informatica, Università di Padova 16/21

Università degli Studi di Padova Risorse protette

Restrizioni d'uso - 2

- ❑ La rilevazione di situazioni erronee comporta il completamento del programma con eccezione `Program_Error`
- ❑ In caso di mancata rilevazione, il programma può entrare in stallo
- ❑ Sono potenzialmente bloccanti
 - Comandi `select`, `accept`, `entry call`, `delay [until]`, `new`
 - Qualunque chiamata a sottoprogramma che sia al suo interno potenzialmente bloccante

Corso di Laurea Specialistica in Informatica, Università di Padova 17/21

Università degli Studi di Padova Risorse protette

Elaborazione e finalizzazione

- ❑ L'elaborazione di una risorsa protetta avviene quando il suo ambito (*scope*) la richiede (ne dichiara un'istanza)
- ❑ 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`

Corso di Laurea Specialistica in Informatica, Università di Padova 18/21

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 RP aiuta a realizzare **protocolli d'accesso** ai servizi piuttosto che i servizi veri e propri

Corso di Laurea Specialistica in Informatica, Università di Padova

19/21

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 Write;
    
```

```

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;
    
```

Corso di Laurea Specialistica in Informatica, Università di Padova

20/21

Università degli Studi di Padova

Risorse protette

Stati d'esecuzione di processo

Corso di Laurea Specialistica in Informatica, Università di Padova

21/21