


 Risorse protette

## Risorse protette



Anno accademico 2007/8  
 Corso di Sistemi Concorrenti e Distribuiti  
 Tullio Vardanega, [tullio.vardanega@math.unipd.it](mailto:tullio.vardanega@math.unipd.it)


Corso di Laurea Specialistica in Informatica, Università di Padova 1/22

 Risorse protette

## Limiti del modello base

- È desiderabile ottenere **mutua esclusione e sincronizzazione condizionale senza dover necessariamente impiegare un processo *server***
- È anche desiderabile che l'astrazione proposta abbia **potenza espressiva non inferiore al modello del processo *server***
- I modelli base 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 2/22

 Risorse protette

## Mutua esclusione – 1

- **Requisiti**
  1. **Accesso in scrittura in mutua esclusione**
  2. **Accesso in sola lettura potenzialmente concorrente**
  3. **Distinzione strutturale tra operazioni di sola lettura e 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 3/22

 Risorse protette

## Mutua esclusione – 2

- **Vantaggi rispetto all'impiego di un processo *server***
  - **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 ambiente (*scope*) cessa di esistere
    - L'onere è tutto e solo a carico dei processi cliente
- **Potenziali controindicazioni**
  - **Possibili limitazioni nella logica dell'agente di controllo**


Corso di Laurea Specialistica in Informatica, Università di Padova 4/22

 Risorse protette

## Sincronizzazione condizionale – 1

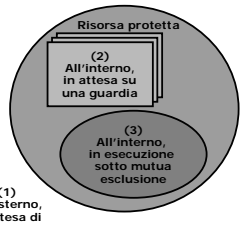
- **Requisiti**
  - **Possibilità per un processo cliente di sincronizzarsi con una condizione (legata a uno stato o dato di servizio) invece che con un altro processo**
  - **Possibilità per un processo cliente di sospendersi in attesa che la condizione attesa si verifichi**
- **Possibile soluzione**
  - **Punto di accesso (*entry*) offerto da una risorsa protetta, sul quale applica una guardia Booleana**
    - La guardia opera come una precondizione sull'esecuzione del servizio
  - **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

Corso di Laurea Specialistica in Informatica, Università di Padova 5/22

 Risorse protette

## Sincronizzazione condizionale – 2

**Il modello a "guscio d'uovo"**



- **2 livelli di protezione**
  - **Esecuzione in mutua esclusione (stato: 3)**
  - **Attesa su guardia Booleana di precondizione, senza rischi di *starvation* (stato: 2)**
- **Valutazione della guardia in mutua esclusione**
  - **Nel passaggio da (stato: 1) verso (stato: 2) o (stato: 3)**

Corso di Laurea Specialistica in Informatica, Università di Padova 6/22

Risorse protette

### Esempio: un contenitore vincolato

```

Buffer_Size : constant Positive := 5;
type Index is mod Buffer_Size; -- tipo ad incremento modulare
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; -- nessun rischio di overflow
    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; -- nessun rischio di overflow
    Buffer(Last) := Item;
    in_Buffer := in_Buffer + 1;
  end Put;
end Bounded_Buffer;
    
```

7/22

Corso di Laurea Specialistica in Informatica, Università 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 di RP 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 di RP viene rivalutata
  - A ogni richiesta d'accesso la cui guardia abbia una componente che possa essere cambiata dall'ultima valutazione
    - Dunque **non** a seguito di una chiamata di una funzione protetta!
  - A 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

8/22

Corso di Laurea Specialistica in Informatica, Università di Padova

Risorse protette

### Sincronizzazione condizionale – 4

- ❑ Una RP è detta "in uso per lettura" (**read lock**) quando  $\geq 1$  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 che esse siano soggette a competizione
  - Ciò **non può** però avvenire **indirettamente**, ossia quando la chiamata venisse effettuata da un sottoprogramma esterno alla risorsa, ma chiamato dall'interno di essa
    - Tipica situazione **erronea** che induce rischio di stallo
- ❑ In entrambi i casi (**R, R/W lock**) ogni altra chiamata viene trattenuta **all'esterno** della RP

9/22

Corso di Laurea Specialistica in Informatica, Università 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 o **entry**, la chiamata viene **differita** fin quando vi siano processi attivi all'interno di RP
3. Se RP è sotto "**R/W lock**", qualunque chiamata viene **differita** fin quando vi siano processi attivi all'interno di RP i quali abbiano requisiti di accesso potenzialmente in conflitto con tale 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

10/22

Corso di Laurea Specialistica in Informatica, Università di Padova

Risorse protette

### Protocollo d'accesso – 2

8. Se la chiamata è a una **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 (**time out**) e si passa al punto 10
10. Viene **rivalutata** ciascuna guardia che abbia chiamate in attesa, 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

11/22

Corso di Laurea Specialistica in Informatica, Università di Padova


Risorse protette

### Protocollo d'accesso – 3

- ❑ L'aspetto più delicato del protocollo riguarda i punti 10.–11.
  - Rivalutazione delle guardie
- ❑ Le guardie vengono rivalutate a seguito dell'esecuzione di procedure o **entry**, oltre che al punto di chiamata (8.)
- ❑ Per ognuna di quelle risultano aperte si esegue il corpo dell'**entry** corrispondente
  - Il protocollo **non** deve specificare quale processo debba prendersi carico dell'esecuzione!
  - Potrebbe essere il processo che ha emesso la chiamata oppure quello la cui azione **abbia aperto la guardia**
    - L'unica differenza che deriva dalla scelta è nell'ordine di esecuzione che ne risulta
    - Quale scelta comporta l'onere inferiore?
- ❑ Le clausole temporali di selezione poste dal chiamante sono valutate **solo** a partire da quando la chiamata venga effettivamente accodata

12/22

Corso di Laurea Specialistica in Informatica, Università di Padova




Risorse protette

## Controllo d'accodamento – 1

- ❑ Ogni punto d'accesso a canale tipato possiede un attributo predefinito `Count`
  - Funzione che ritorna il numero di processi in quel momento in coda sull'*entry*
  - Sia per il *server* che per la risorsa protetta
- ❑ Per questo motivo anche l'accodamento di chiamata richiede *"write lock"* sulla risorsa!
- ❑ L'uso dell'attributo `Count` nell'espressione Booleana di una guardia ne comporta l'obbligo di rivalutazione a ogni nuovo accodamento di chiamata

Corso di Laurea Specialistica in Informatica, Università di Padova

13/22



Risorse protette

## Controllo d'accodamento – 2

Esempio: visita a 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_InCount = N or Open is
begin
if Let_InCount = 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 I chiamata accodata modificherà `Open` così che la guardia resti aperta fino a che sia stato rilasciata la V chiamata accodata, la quale chiuderà di nuovo la guardia

Vediamone l'esecuzione ...

Corso di Laurea Specialistica in Informatica, Università di Padova

14/22



Risorse protette

## Controllo d'accodamento – 3

- ❑ L'attributo `Count` fornisce grande potenza espressiva
- ❑ È bene però sapere che usarlo nell'espressione di una guardia può causarne valutazioni ripetute senza soluzione di continuità
  - Alla chiamata del punto d'accesso
  - Al possibile accodamento della richiesta in presenza di guardia chiusa
    - A ogni chiamata nel caso dell'esempio di pagina 14!

Corso di Laurea Specialistica in Informatica, Università di Padova

15/22



Risorse protette

## Restrizioni d'uso – 1

- ❑ Sul piano metodologico l'esecuzione entro una RP dovrebbe essere il più breve possibile
  - Anche quelle all'interno di una sincronizzazione tra cliente e servente
- ❑ Per limitarne la durata, si considerano erronee le azioni **potenzialmente bloccanti** effettuate entro azioni protette
  - Vogliamo proteggere la logica implicita del modello
  - Nel caso del *server* la protezione è a carico del programmatore

Corso di Laurea Specialistica in Informatica, Università di Padova

16/22




Risorse protette

## Restrizioni d'uso – 2

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

Corso di Laurea Specialistica in Informatica, Università di Padova

17/22



Risorse protette

## Elaborazione e finalizzazione

- ❑ L'elaborazione di una risorsa protetta avviene quando il suo ambito (*scope*) la richiede
  - Quando 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

18/22

Risorse protette

## Ordinamento preferenziale – 1

□ **Il solo vincolo di mutua esclusione può risultare troppo rigido**

- Quando il sistema richiedesse ordinamento preferenziale dei servizi
  - Come rendere le scritture ('write lock') preferibili a letture ('read lock')?
- Quanto l'esecuzione di scritture e letture sulla risorsa condivisa fosse bloccante e quindi non effettuabile entro risorsa protetta
  - Per esempio su un dispositivo *hardware* esterno

□ **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

19/22

Risorse protette

## Ordinamento preferenziale – 2

```
protected Access_Control is
entry Start_Read;
procedure Stop_Read;
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.Start_Write;
  ... -- actual write
  Access_Control.Stop_Write;
end Write;
```

```
protected body Access_Control is
entry Start_Read when [not Writers] and
  [Start_Write_Count = 0] is
begin
  Readers := Readers + 1;
end Start_Read;
-- Preferenza a scrittura

procedure Stop_Read is
begin
  Readers := Readers - 1;
end Stop_Read;

entry Start_Write when [not Writers] and
  [Readers = 0] is
begin
  Writers := True;
end Start_Write;
-- Mutua esclusione con scritture e letture

procedure Stop_Write is
begin
  Writers := False;
end Stop_Write;
end Access_Control;
```

Corso di Laurea Specialistica in Informatica, Università di Padova

20/22

Risorse protette

## Ordinamento preferenziale – 3

□ **Il protocollo d'accesso realizzato tramite RP consente**

- Scritture in mutua esclusione (guardia G2)
  - Come se fossero realizzate all'interno di una RP
- Preferenza a scritture su letture (guardia G1)

□ **Naturalmente quando la risorsa è esterna alla RP si ha serio rischio di stallo**

- La terminazione erronea di un lettore o scrittore impedisce il rilascio della RP e blocca il protocollo

Corso di Laurea Specialistica in Informatica, Università di Padova

21/22

Risorse protette

## Stati d'esecuzione di processo

```

graph TD
    InAttivazione[In attivazione] --> InEsecuzione[In esecuzione]
    InEsecuzione --> Completato[Completato]
    InEsecuzione --> AttivazioneFigli[Attivazione figli]
    InEsecuzione --> Sospensione[Sospensione temporanea]
    InEsecuzione --> Sincronizzazione[Sincronizzazione]
    InEsecuzione --> AccodamentoCliente[Accodamento presso punto d'accesso (cliente)]
    InEsecuzione --> ConTimeOut[Con time-out]
    InEsecuzione --> ConTimeOut[Con time-out]
    InEsecuzione --> AttesaRichiesta[Attesa di richiesta d'accesso (servente)]
    InEsecuzione --> AttesaSelettiva[Attesa selettiva]
    InEsecuzione --> TerminazioneDipendenti[Terminazione dipendenti]
    InEsecuzione --> AccodamentoProtetto[Accodamento presso punto d'accesso protetto]
    
```

Corso di Laurea Specialistica in Informatica, Università di Padova

22/22