Unità di apprendimento 2

Comunicazione e sincronizzazione

Unità di apprendimento 2

Lezione 1

La comunicazione tra processi

In questa lezione impareremo:

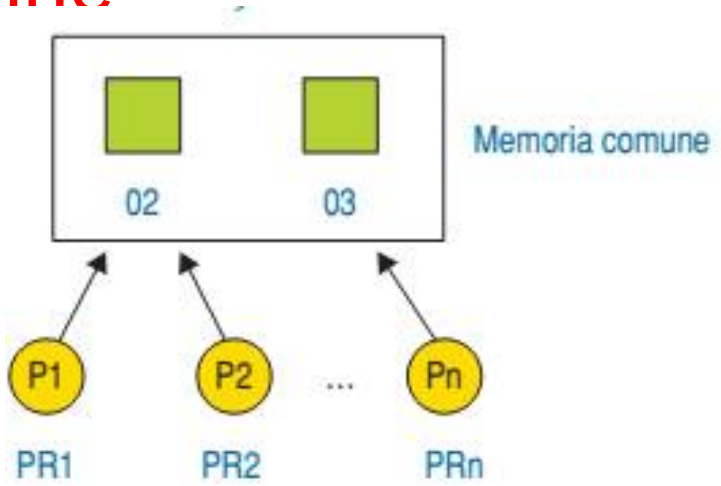
- **il modello ad ambiente globale o a memoria condivisa**
- **il modello ad ambiente locale o a scambio di messaggi**

Comunicazione: modelli software e hardware

- Possiamo individuare due modelli di interazione concorrente a prescindere dalla soluzione hardware, cioè sia che stiamo analizzando una architettura distribuita oppure una situazione di multitask su macchina SISM:
  - modello a memoria comune (ambiente globale, global environment);
  - modello a scambio di messaggi (ambiente locale, message passing).

Modello a memoria comune
(ambiente globale, global environment)

- Il modello a memoria comune trova naturale impiego nelle architetture in cui esiste un'unica memoria comune a tutti i processi (o processori)
- Ad esempio su macchine monoprocessore con processi multitasking



Comunicazione: modelli software e hardware

- **Allocazione delle risorse ai processi**
- Il modello a memoria comune è il solo caso nel quale possono verificarsi problemi per l'accesso
  - due (o più) processi che ne potrebbero richiedere l'attribuzione di memoria contemporaneamente
- Il sistema operativo associa a ogni risorsa un apposito *gestore di risorsa* (o *allocatore*)

Comunicazione: modelli software e hardware

- L'allocatore ha i seguenti compiti:
  - deve mantenere aggiornato lo *stato di allocazione della risorsa*;
  - deve *fornire i meccanismi* ai processi che hanno il diritto di utilizzare tale risorsa di accedervi, prenderne possesso, operare su di essa e alla fine del suo utilizzo di "liberarla" per gli altri processi;
  - deve *implementare la strategia di allocazione* della risorsa definendo a quale processo e per quanto tempo assegnare la risorsa.

Comunicazione: modelli software e hardware

In generale un allocatore si interfaccia con i processi mediante due procedure:

- al momento in cui un processo ha bisogno di una risorsa fa una richiesta di assegnazione (*acquisizione*) mediante la prima procedura;
- la seconda viene chiamata dal processo che sta utilizzando la risorsa quando termina di averne bisogno e intende "renderla libera" restituendola al sistema operativo (*rilascio*).

ESEMPIO

- un processo che necessita di stampare un documento
- Il gestore della risorsa è lo spooler di stampa che all'atto di una richiesta la pone in una coda di attesa e la soddisfa
- quando un processo termina di stampare e lascia libera la stampante.

Comunicazione: modelli software e hardware

- **Tipologie di allocazione delle risorse nel modello ad ambiente globale**

	Risorse dedicate (visibili in ogni istante da un solo processo)	**Risorse condivise** (visibili in ogni istante anche da più processi contemporaneamente)
risorse allocate staticamente	risorse private	risorse comuni
risorse allocate dinamicamente	risorse comuni	risorse comuni

Per ogni risorsa il relativo gestore definisce istante per istante i processi che hanno il diritto di operare su di essa.

Comunicazione: modelli software e hardware

Le risorse allocate staticamente vengono definite prima che il programma inizi la propria esecuzione e quindi il loro gestore è il programmatore.

- Nel caso di risorse dedicate non è necessario nessun controllo da parte del programmatore per quanto riguarda la sincronizzazione dato che queste sono di utilizzo esclusivo di un singolo processo e vengono assegnate e gestite dal sistema operativo.
- Nel caso di risorse condivise il programmatore stabilisce le regole di visibilità e quindi quali processi possono operare sui dati comuni, in quali istanti possono accedere alla risorsa, definendo le modalità di sincronizzazione.

Comunicazione: modelli sw - hw

ESEMPIO - regole di visibilità

```
int x;
f()
{
   int y;
   y=1;
}
```

- La visibilità di y si estende dal punto di dichiarazione fino alla fine del blocco di appartenenza

```
int x;
g(int y, char z)
{
   int k;
   int l;
   …
}
```

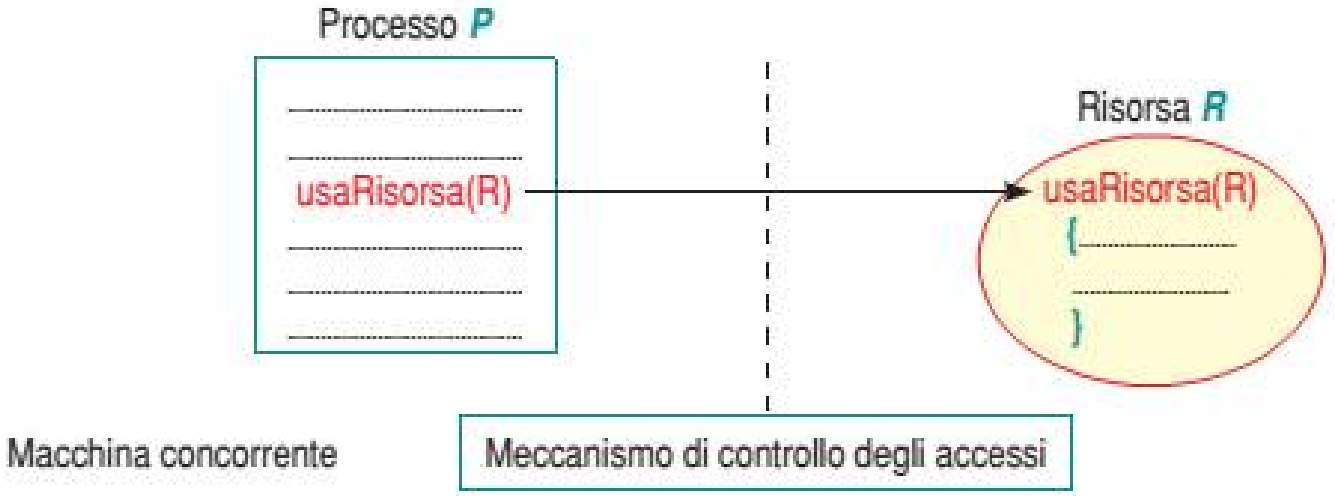
- **y** e **z** locali alla funzione **g**, con visibilità nel blocco racchiuso da parentesi graffe
- **k** e **l** hanno la stessa visibilità

```
f(int x)
{
   int x;
}
```

- Errata! Si tenta di definire due volte la variabile locale x nello stesso blocco

Comunicazione: modelli software e hardware

Gli accessi a una risorsa condivisa devono avvenire in modo non divisibile: le funzioni che utilizzano un dato condiviso (sezioni critiche) devono essere programmate utilizzando i meccanismi di sincronizzazione offerti dal linguaggio di programmazione e supportati dalla macchina concorrente.



Comunicazione: modelli software e hardware

MUTUA ESCLUSIONE

L'accesso a una risorsa si dice mutuamente esclusivo se a ogni istante, al massimo un processo può accedere a quella risorsa.

- Vediamo nelle diverse situazioni come interagiscono i processi nella situazione di condivisione delle risorse:
  - quando competono
  - quando cooperano
  - quando interferiscono in situazioni di allocazione statica e dinamica

Comunicazione: modelli software e hardware

- **Competizione**

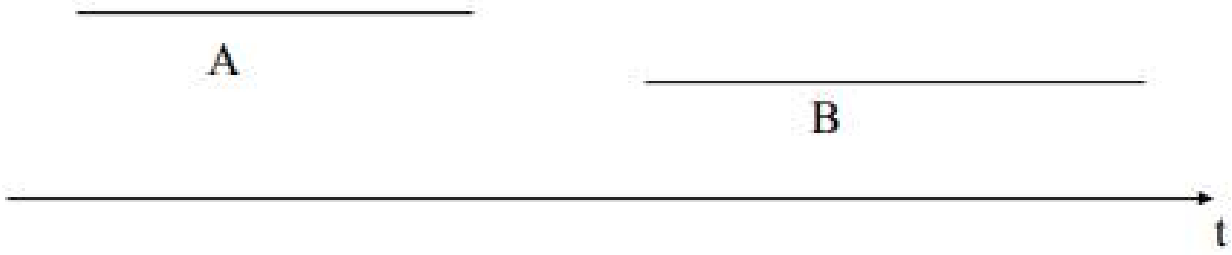
La competizione è una interazione tra processi prevedibile e NON desiderata.

- Risorse condivise e allocate staticamente:
  - la competizione avviene al momento dell'accesso alla risorsa: è necessario garantire l'accesso esclusivo (mutua esclusione).
- Risorse dedicate e allocate dinamicamente:
  - la responsabilità di gestione è demandata al gestore e la competizione tra processi avviene al momento della richiesta di utilizzo

Comunicazione: modelli sw – hw
mutua esclusione

MUTUA ESCLUSIONE

• **Il problema della mutua esclusione** nasce quando più di un processo alla volta può aver accesso a variabili comuni.
• La regola di mutua esclusione impone che le operazioni con le quali i processi accedono alle variabili comuni *non si sovrappongano nel tempo*.
• Nessun vincolo è imposto *sull'ordine* con il quale le operazioni sulle variabili vengono eseguite.



Comunicazione: modelli sw – hw
mutua esclusione

Esempi di mutua esclusione

Esempio 1

• Due processi P1 e P2 hanno accesso ad una struttura *organizzata a* **pila** per prelevare ed inserire dati.

• La struttura dati è rappresentata da un *vettore stack* i cui elementi costituiscono i singoli dati e da una *variabile top* che indica la posizione dell'ultimo elemento contenuto nella pila.

• I processi utilizzano le operazioni *Inserimento e Prelievo* per depositare e prelevare i dati dalla pila.

Comunicazione: modelli sw – hw
mutua esclusione

```
var  stack: array(1..n) of T
       top:integer initial (0);
procedure Inserimento (y:T);
    begin
      top:=top+1;
      stack(top):=y;
    end
procedure Prelievo(var x:T);
    begin
      x:= stack(top);
      top:=top-1;
    end
```

Comunicazione: modelli sw – hw
mutua esclusione

• L'esecuzione contemporanea di queste operazioni da parte dei processi può portare ad un uso scorretto della risorsa.

• Possibile sequenza di esecuzione delle due operazioni:

T0:	*top:= top+1*;	(P1)
T1:	*x:= stack(top)*;	(P2)
T2:	*top:= top-1*;	(P2)
T3:	*stack(top):= y*;	(P1)

• Viene assegnato a x un valore *non definito* e l'ultimo valore valido contenuto nella pila *viene cancellato* dal nuovo valore di y.

•Analogamente si avrebbe nel caso di esecuzione contemporanea di una qualunque delle due operazioni da parte dei due processi.

Comunicazione: modelli sw – hw
mutua esclusione

Ne caso di risorse dedicate e allocate dinamicamente:

la responsabilità di gestione è demandata al gestore e la competizione tra processi avviene al momento della richiesta di utilizzo indirizzata al gestore stesso che provvederà a gestirle in mutua esclusione.

La soluzione della competizione avviene mediante la sincronizzazione indiretta o implicita gestita dal sistema operativo

Comunicazione: modelli software e hardware

- **Cooperazione**

La cooperazione è una interazione tra processi prevedibile e desiderata dato che è insita nella logica del programma.

- la cooperazione tra processi avviene utilizzando una risorsa condivisa
- un processo (o più processi) scrivono un dato nella risorsa (produttori) e un altro processo (o più processi) leggono successivamente dalla risorsa condivisa (consumatori)

Comunicazione: modelli software e hardware

Cooperazione

Nel caso di risorse dedicate e allocate dinamicamente l'unica possibilità di cooperazione si ha se una risorsa assegnata a un processo viene assegnata a un secondo processo quando il primo termina di utilizzarla: il gestore deve essere al corrente della cooperazione in modo da non azzerare il dato "prodotto" dal primo processo prima che venga "consumato" dal secondo processo.

La soluzione della cooperazione avviene mediante sincronizzazione diretta o esplicita, gestita dal sistema dal programmatore mediante *scambio di informazioni*.

Comunicazione: modelli software e hardware

- **Interferenza**
- L'interferenza è dovuta:
  - alla competizione che avviene tra processi che utilizzano senza le opportune autorizzazioni risorse condivise
  - a una erronea soluzione dei problemi di competizione e di cooperazione

Comunicazione: modelli software e hardware

- **Interferenza - Esempio**

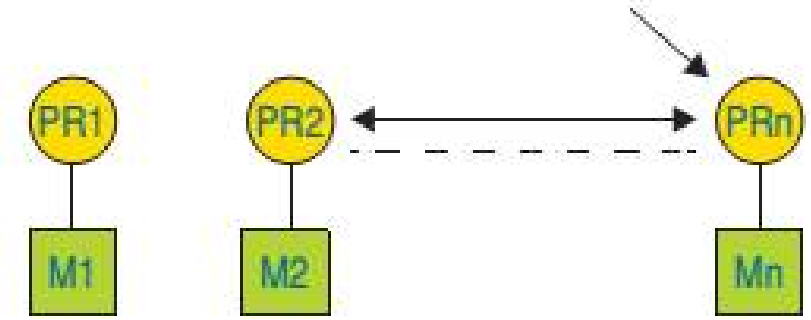
Per esempio possiamo essere in una delle due seguenti situazioni:

- P1 chiama P2 senza necessità e lo rallenta;
- P1 chiama P2 ma al momento sbagliato.

Generalmente i malfunzionamenti sono legati alla velocità relativa dei processi e quindi i risultati sono "dipendenti dal tempo": di conseguenza è molto complesso effettuare il debug.

Modello a scambio di messaggi (ambiente locale, message passing)

- Ogni processo può accedere esclusivamente alle risorse allocate nella propria memoria locale
- Questa non può essere modificata direttamente dagli altri processi



Modello a scambio di messaggi (ambiente locale, message passing)

- Esistono due possibili modalità:
  - utilizzare linguaggi che prevedono *costrutti espliciti* per realizzare lo scambio di messaggi, come il CSP (Communicating Sequential Processes) proposto da Tony Hoare
  - utilizzare la "*chiamata di procedura remota*", come il DP (Distributed Processes), proposto da Brinch Hansen

Modello a scambio di messaggi (ambiente locale, message passing)

Un semplice linguaggio:CSP

- Il linguaggio CSP è una architettura parallela caratterizzata da:
  - Scambio di messaggi tra processi
  - Possibilità per i thread di utilizzare memoria condivisa
- Lo scambio di messaggi avviene tramite **Canali**
- I canali permettono:
  - Uno scambio di valori tra processi
  - La sincronizzazione di processi
- Il numero dei canali è fisso:
  - Non è possibile creare o cancellare un canale durante l'esecuzione di un programma.
  - È possibile avere canali dedicati a singoli processi

Modello a scambio di messaggi (ambiente locale, message passing)

Sintassi di CSP

Commands

$c ::=$	**skip** | **abort** | $X := a \mid c_0 ; c_1$	*standard commands*
|	$\alpha!a$	send value on channel (output)
|	$\alpha?X$	receive value on channel (input)
|	$c_0 \parallel c_1$	parallel composition (locations disjoint)
|	$c \setminus \alpha$	restriction: hide channel α
|	**if** g **fi**	alternative
|	**do** g **od**	iteration

Guarded Commands

$g ::=$	$b \rightarrow c$	boolean guard
|	$b \wedge \alpha?X \rightarrow c$	boolean receive guard
|	$b \wedge \alpha!a \rightarrow c$	boolean send guard
|	$g_0 \,\square\, g_1$	alternative

–Based on IMP: $a \in AExp$, $b \in BExp$, $X \in Loc$

Modello a scambio di messaggi (ambiente locale, message passing)

- Classificazione dei modelli a scambio di messaggi:
  - comunicazione asincrona la comunicazione da parte del processo mittente avviene senza che questo rimanga in attesa di una risposta da parte del processo destinatario
  - comunicazione sincrona lo scambio di informazioni può avvenire solo se mittente e destinatario sono pronti a “parlarsi” e quindi è necessario che si sincronizzino

Modello a scambio di messaggi (ambiente locale, message passing)

- Nel caso di comunicazione sincrona si hanno due tipologie di incontro o sincronismo “rendez-vous”:
  - stretto: se si limita alla trasmissione di un messaggio dal mittente al destinatario;
  - esteso: se il destinatario, una volta ricevuto il messaggio, deve inviare una risposta al mittente

Modello a scambio di messaggi (ambiente locale, message passing)

- Distinguiamo la comunicazione in:
  - asimmetrica: il mittente nomina esplicitamente il destinatario ma questo non nomina esplicitamente il mittente;
  - simmetrica: entrambi si nominano in modo esplicito.
- Possiamo avere due classiche situazioni:
  - comunicazione di tipo simmetrico e sincrono a rendez-vous stretto tipico del CSP
  - comunicazione di tipo asimmetrico e sincrono a rendez-vous esteso tipico del DP

Modello a scambio di messaggi (ambiente locale, message passing)

- **Modello client-server**
- Ogni risorsa del sistema è accessibile a un solo processo che prende il nome di processo servitore (o server)
- quando un processo deve utilizzarla (processo cliente) non può accedervi direttamente ma deve chiedere al processo server di effettuare lui stesso le operazioni desiderate sulla risorsa e di comunicargli successivamente l'esito delle elaborazioni.

SERVITORE/CLIENTE

Servitore: entità computazionale in grado di eseguire una specifica prestazione per altre entità (in grado cioè di offrire un servizio).
Cliente: entità computazionale che richiede a un servitore l'esecuzione di una specifica prestazione.