Interazione tra Processi

Processi interagenti

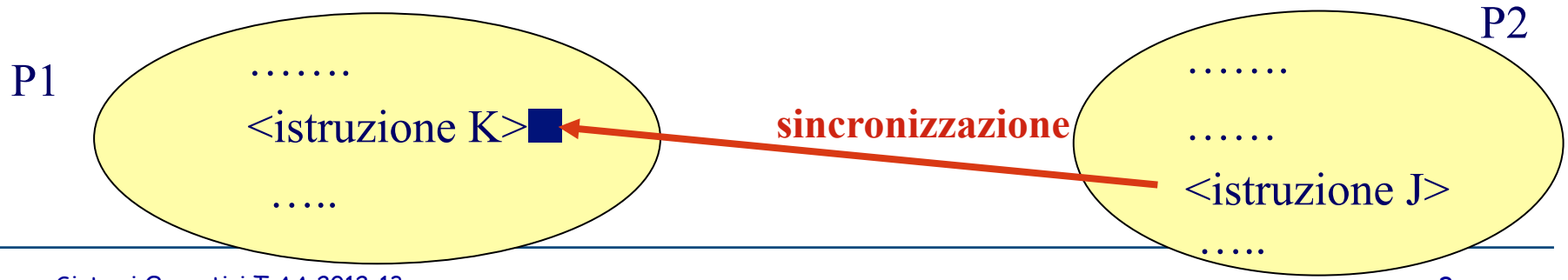
Classificazione:

- **processi interagenti/indipendenti:**
  - due processi sono interagenti se l'esecuzione di un processo è in alcun influenzata dall'esecuzione dell'altro processo o viceversa.
  - due processi sono indipendenti se l'esecuzione di ognuno non è in alcun modo influenzata dall'altro.

- **processi interagenti:**
  - **cooperanti**: i processi interagiscono volontariamente per raggiungere obiettivi comuni (fanno parte della stessa applicazione)
  - **in competizione:** i processi, in generale, non fanno della stessa applicazione, ma interagiscono indirettamente, per l'acquisizione di risorse comuni.

Processi Interagenti

- L'interazione puo` avvenire mediante due **meccanismi**:
  - ✓ **Comunicazione**: scambio di informazioni tra i processi interagenti.
  - ✓ **Sincronizzazione**: imposizione di vincoli temporali sull'esecuzione dei processi.
    Ad esempio, l'istruzione K del processo P1 puo` essere eseguita soltanto dopo l'istruzione J del processo P2



Processi Interagenti

- **Realizzazione dell'interazione**:dipende dal modello di processo:
  - **modello ad ambiente locale** (**processo pesante**): non c'è condivisione di variabili
    - la comunicazione avviene attraverso **scambio di messaggi**
    - la sincronizzazione avviene attraverso **scambio di eventi** (*segnali*)
  - **modello ad ambiente globale** (***thread***): più processi possono condividere lo stesso spazio di indirizzamento => possibilità di condividere variabili
    - variabili condivise + strumenti di sincronizzazione (es.*semafori*)

Processi interagenti mediante scambio di messaggi

Facciamo riferimento al modello ad ambiente locale:

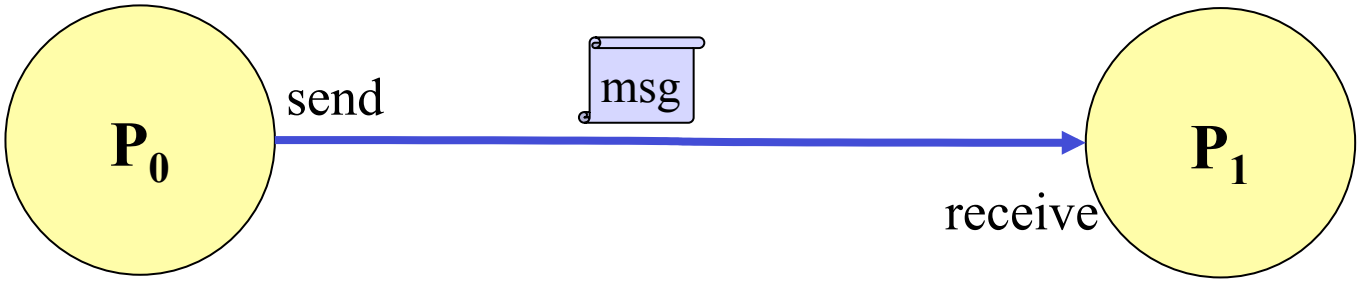
- non vi è memoria condivisa
- i processi possono interagire (*cooperano/competono*)
  - mediante scambio di messaggi: *comunicazione*
  - mediante *scambio di eventi (o segnali): sincronizzazione*

➢ Il Sistecomunicazioma Operativo offre meccanismi a supporto della ne tra processi (*Inter Process Communication*, o IPC).

Operazioni Necessarie:

- *send*: spedizione di messaggi da un processo ad altri
- *receive*: ricezione di messaggi

Scambio di messaggi



Lo scambio di messaggi avviene mediante un **canale di comunicazione** tra i due processi

Caratteristiche del canale:

- monodirezionale, bidirezionale
- uno-a-uno, uno-a-molti, molti-a-uno, molti-a-molti
- capacità
- modalità di creazione: automatica, non automatica

Meccanismi di comunicazione tra processi

Aspetti caratterizzanti:

- caratteristiche del canale
- caratteristiche del messaggio:
  - » dimensione
  - » tipo
- tipo della comunicazione:
  - » diretta o indiretta
  - » simmetrica o asimmetrica
  - » bufferizzata o no
  - » ...

Naming

In che modo viene specificata la destinazione di un messaggio?

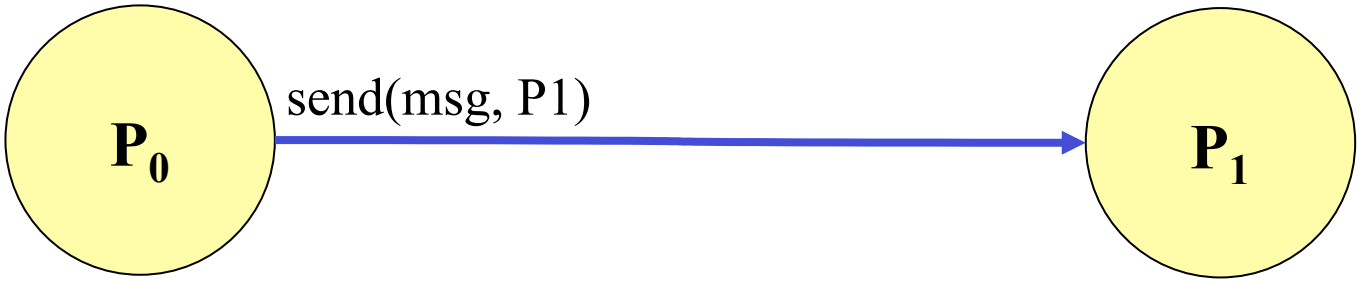
- **Comunicazione diretta**: al messaggio viene associato l'identificatore del processo destinatario (naming esplicito)

  send(Proc, msg)

- **Comunicazione indiretta**: il messaggio viene indirizzato a una mailbox (contenitore di messaggi) dalla quale il destinatario preleverà il messaggio:

  send(Mailbox, msg)

Comunicazione diretta



- Il canale è creato automaticamente tra i due processi che devono *conoscersi* reciprocamente:
  - canale punto-a-punto
  - canale bidirezionale:

    p0: send(query, P1);

    p1: send(answ, P0)
  - per ogni coppia di processi esiste un solo canale(<P0, P1>)

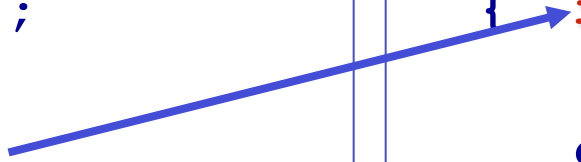
Esempio: Produttore & Consumatore

Processo produttore P:

```
pid C =....;
main()
{ msg M;
   do
   { produco(&M);
     ...
     send(C, M);
   }while(!fine);
}
```

Processo consumatore C:

```
pid P=....;
main()
{ msg M;
   do
   { receive(P, &M);
     ...
     consumo(M);
   }while(!fine);
}
```



Comunicazione simmetrica:

- il destinatario fa il ***naming*** esplicito del mittente

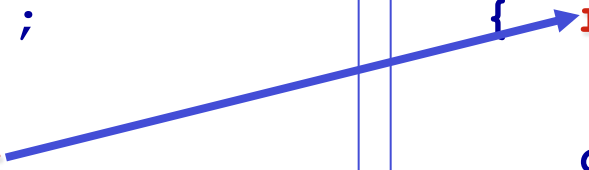
Comunicazione asimmetrica

Processo produttore P:

```
....
main()
{ msg M;
   do
   { produco(&M);
     ...
     send(C, M);
   }while(!fine);
}
```

Processo consumatore C:

```
....
main()
{ msg M; pid id;
   do
   { receive(&id, &M);
     ...
     consumo(M);
   }while(!fine);
}
```



Comunicazione asimmetrica:

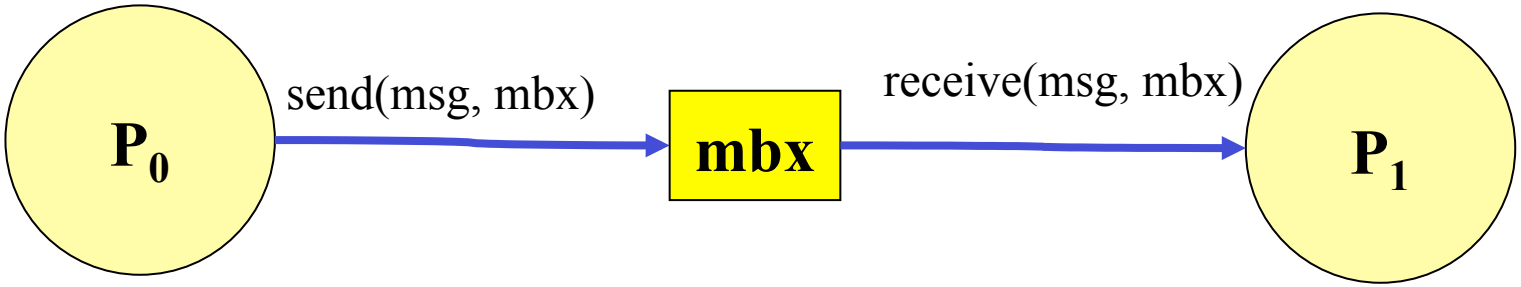
- il destinatario non è obbligato a conoscere l'identificatore del mittente: la variabile **id** raccoglie l'identificatore del mittente.

Comunicazione diretta

Problema:

- **scarsa modularità**:
  - la modifica del nome di un processo implica la revisione di tutte le operazioni di comunicazione
  - difficoltà di riutilizzo
  - utilità di servizi di directory (name server)

Comunicazione indiretta



- I processi cooperanti non sono tenuti a conoscersi reciprocamente e si scambiano messaggi depositandoli/prelevandoli da una mailbox *condivisa.*
- La **mailbox** è una risorsa astratta condivisibile da più processi che funge da contenitore dei messaggi.

Comunicazione indiretta

Proprietà:

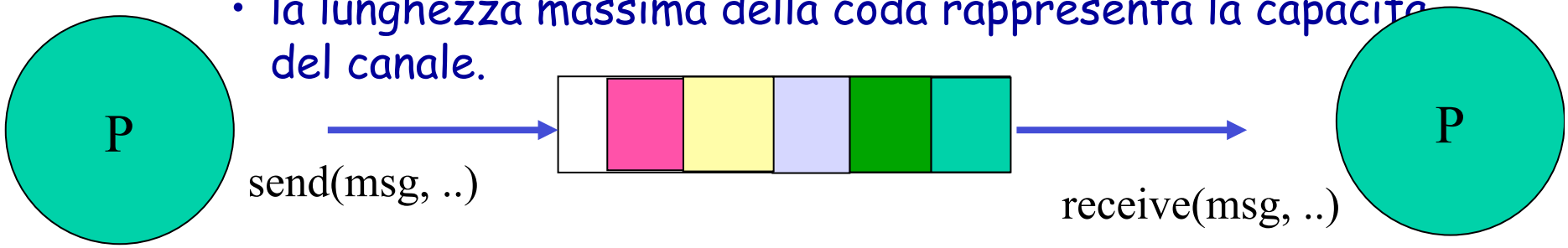
- il canale di comunicazione è rappresentato dalla mailbox (non viene creato automaticamente)
- il canale può essere associato a più di 2 processi:
  - » mailbox di sistema: molti-a-molti (come individuare il processo destinatario di un messaggio?)
  - » mailbox del processo destinatario (**porta**): molti-a-uno
- canale bidirezionale:

  p0: send(query, mbx)

  p1: send(answ, mbx)
- per ogni coppia di processi possono esistere più canali (uno per ogni mailbox condivisa)

Buffering del canale

- Ogni canale di comunicazione è caratterizzato da una **capacità**: numero dei messaggi che è in grado di contenere contemporaneamente.
- Gestione secondo politica **FIFO**:
  - i messaggi vengono posti in una coda in attesa di essere ricevuti
  - la lunghezza massima della coda rappresenta la capacità del canale.



Buffering del canale

- **Capacità nulla**: non vi è accodamento perchè il canale non è in grado di gestire messaggi in attesa
  - processo mittente e destinatario devono **sincronizzarsi** all'atto di spedire (send) / ricevere (receive) il messaggio: comunicazione **sincrona** o ***rendez vous***
  - ***send e receive*** possono essere **sospensive**:

Buffering del canale

- **Capacità non nulla (limitata)**: esiste un limite N alla dimensione della coda:
  - se la coda **non è piena**, un nuovo messaggio viene posto in fondo
  - se la coda **è piena**: la send è sospensiva
  - se la coda **è vuota**: la receive può essere sospensiva

- [**Capacità illimitata**: lunghezza della coda teoricamente infinita: non c'è possibilità di sospensione.]

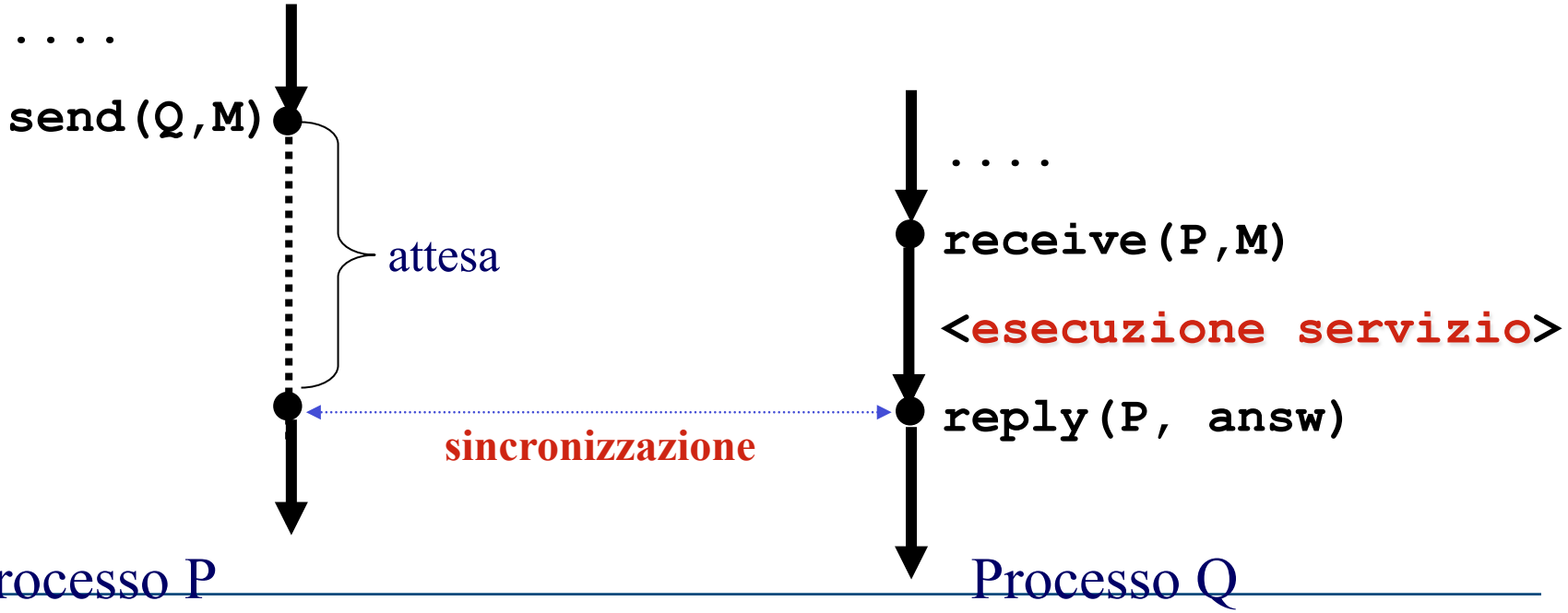
Send sincrona/asincrona

Quindi, a seconda della capacità del canale, la send può essere sospensiva o no:

- **Canale a Capacità nulla: Send sincrona**
  - Se il destinatario non è pronto per ricevere il messaggio, il mittente attende.
- **Canale a Capacità non nulla: Send asincrona**
  - La send non è sospensiva: il mittente deposita il messaggio nel canale e continua la sua esecuzione.

Remote Procedure Call (RPC)

- È un tipo di comunicazione sincrona, in cui il mittente si sospende fino a che il destinatario non restituisce una risposta (***reply***) al messaggio inviato:
  - il messaggio potrebbe richiedere l'esecuzione di un **servizio**



Caratteristiche della comunicazione tra processi Unix

- Due meccanismi di comunicazione:
  - **pipe**: comunicazione locale (nell'ambito della stessa gerarchia di processi)
  - **socket**: comunicazione in ambiente distribuito (tra processi in esecuzione su nodi diversi di una rete)
- Pipe:comunicazione
  - » **indiretta** (senza naming esplicito)
  - » canale **unidirezionale** molti-a-molti
  - » **bufferizzata** (capacità limitata): possibilità di sospensione sia per mittenti che per destinatari.

Sincronizzazione tra processi

Si è visto che due processi possono interagire per:

- **cooperare**: i processi interagiscono allo scopo di perseguire un obiettivo comune
- **competere**:
  - ✓ i processi possono essere logicamente indipendenti,

    ma
  - ✓ necessitano della stessa **risorsa** (dispositivo, file, variabile, ecc.) per la quale sono stati dei vincoli di accesso: ad esempio:
    - » gli accessi dei due processi alla risorsa devono escludersi mutuamente nel tempo (***Mutua Esclusione*** nell'accesso alla risorsa)

➢ In entrambi i casi è necessario disporre di ***strumenti di sincronizzazione***.

Sincronizzazione tra processi

La sincronizzazione permette di imporre vincoli sulle operazioni dei processi interagenti.

Ad Esempio:

Nella cooperazione:

- Per imporre un particolare ordine cronologico alle azioni eseguite dai processi interagenti.
- Per garantire che le operazioni di comunicazione avvengano secondo un ordine prefissato.

Nella competizione:

- Per garantire la mutua esclusione dei processi nell'accesso alla risorsa condivisa.
- Per realizzare politiche di accesso alle risorse condivise

Sincronizzazione tra processi nel modello ad ambiente locale

In questo ambiente non vi e` la possibilita` di condividere memoria:

- Gli accessi alle risorse "condivise" vengono controllati e coordinati dal sistema operativo.
- La sincronizzazione avviene mediante meccanismi offerti dal sistema operativo che consentono la notifica di “eventi” asincroni (privi di contenuto informativo) tra un processo ed altri:
  - **Es. segnali unix**

Sincronizzazione tra processi nel modello ad ambiente globale

- Facciamo riferimento a processi che possono condividere variabili (***modello ad ambiente globale***, o a memoria condivisa) per descrivere alcuni strumenti di sincronizzazione tra processi.

In questo ambiente:

- **cooperazione**: lo scambio di messaggi avviene attraverso strutture dati condivise (ad es., mailbox)
- **competizione**: le risorse sono rappresentate da variabili condivise (ad esempio, puntatori a file)

- In entrambi i casi è necessario sincronizzare i processi per coordinarli nell' accesso alla memoria condivisa:

problema della mutua esclusione