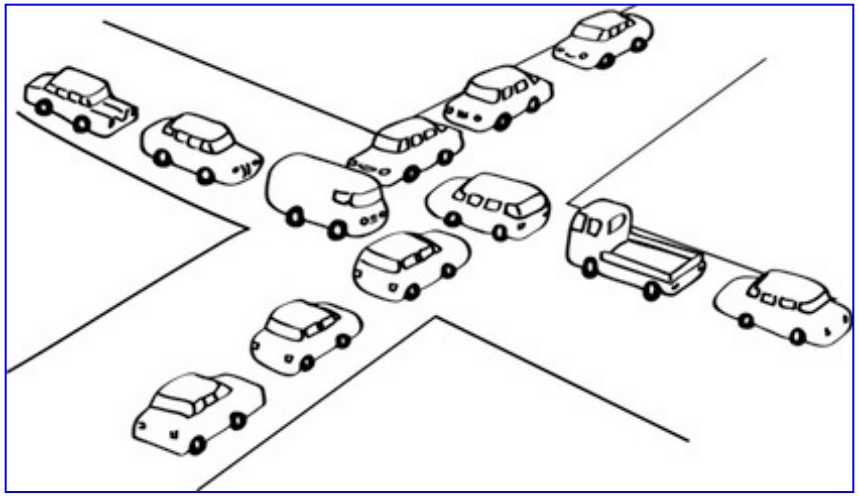
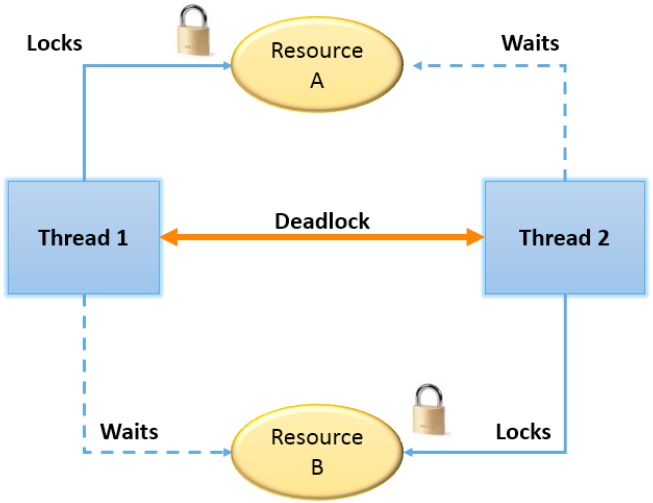
Gestione del deadlock
(prima parte)

Obiettivi

- Descrizione delle situazioni di stallo (*deadlock*) che impediscono il completamento del lavoro a gruppi di processi concorrenti.



- Presentazione dei metodi differenti per prevenire o impedire le situazioni di deadlock tra processi.



Il problema del deadlock

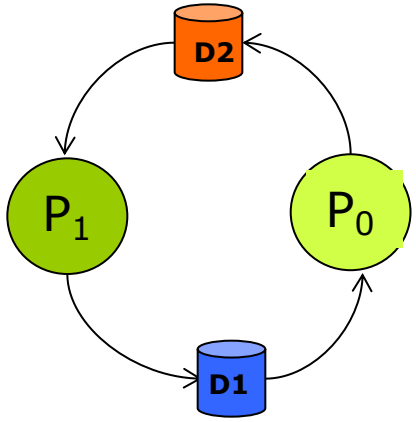
- **Stato di deadlock**: Un insieme di processi bloccati, ognuno dei quali possiede (= ha già acquisito) una risorsa ed è in attesa di acquisire un'altra risolta posseduta (acquisita) da un altro processo dello stesso insieme.

- Esempio:
  - Il sistema comprende due dischi
  - P_1 e P_2 hanno entrambi acquisito uno dei due dischi e hanno bisogno di acquisirne un altro

- Esempio;
  - I semafori *D1* e *D2*, inizializzati ad 1

    P_0 : wait (D1); wait(D2);

    P_1 : wait (D2); wait(D1);



Modello del sistema

- Tipi di risorse R_1, R_2, . . ., R_m

  Cicli di CPU, spazio di memoria, dispositivi di I/O

- Ogni tipo di risorsa R_i ha W_i istanze.

- Ogni processo utilizza una risorsa secondo il seguente schema:
  - **Richiesta (request)**
  - **Uso (use)**
  - **Rilascio (release)**

Condizioni necessarie

Si può verificare un deadlock solo se si verificano contemporaneamente le seguenti condizioni:

- **Mutua esclusione:** solo un processo per volta può usare una risorsa
- **Possesso e attesa:** un processo in possesso di almeno una risorsa attende di acquisire risorse già in possesso di altri processi
- **Impossibilità di preemption:** una risorsa può essere rilasciata soltanto volontariamente dal processo che la possiede, dopo che il processo ha completato il suo task
- **Attesa circolare:** deve esistere un insieme $\{P_0, P_1, \ldots, P_n\}$ di processi tale che P_0 è in attesa di una risorsa posseduta da P_1, P_1 è in attesa di una risorsa posseduta da P_2, …, P_{n-1} è in attesa di una risorsa posseduta P_n, e P_n è in attesa di una risorsa posseduta P_0

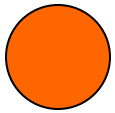
Grafo di allocazione delle risorse

Grafo di allocazione: Un insieme di vertici V e un insieme di archi E.

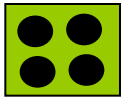
- V è composto da due sottinsiemi:
  - $P = \{P_1, P_2, \ldots, P_n\}$, l'insieme di tutti i processi del sistema
  - $R = \{R_1, R_2, \ldots, R_m\}$, l'insieme di tutti i tipi di risorse del sistema
- **Arco di richiesta** – arco orientato $P_i \rightarrow R_j$
- **Arco di assegnamento** – arco orientato $R_j \rightarrow P_i$

Grafo di allocazione delle risorse

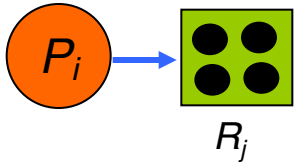
- Processo



- Tipo di risorsa con 4 istanze

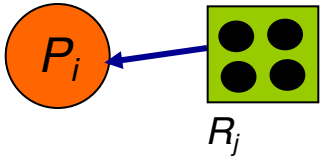


- P_i richiede un' istanza di R_j



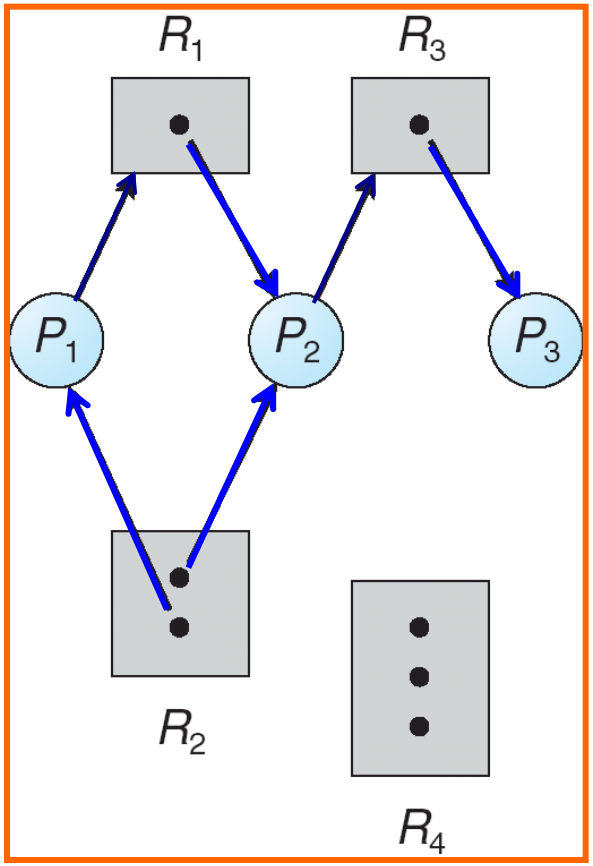
Arco di richiesta

- P_i possiede un' istanza di R_j

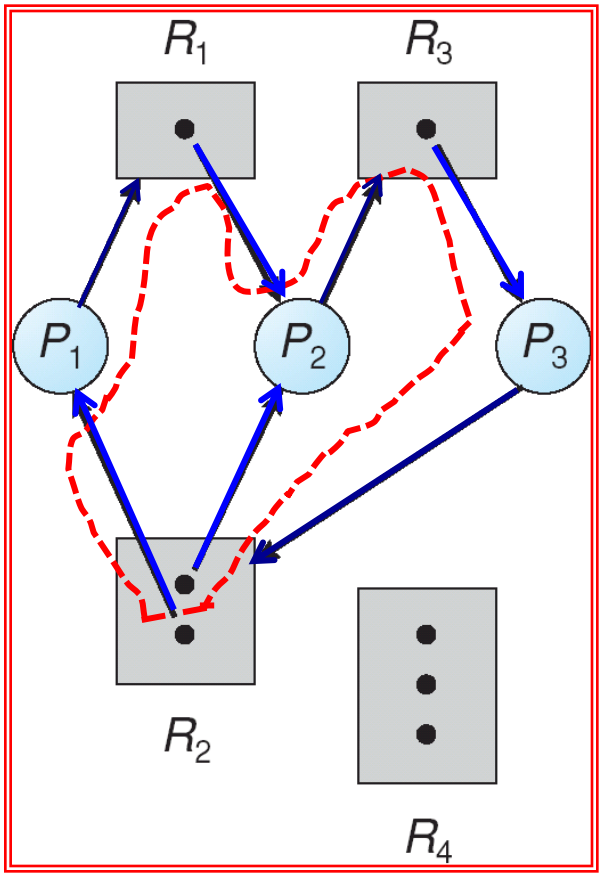


Arco di assegnamento

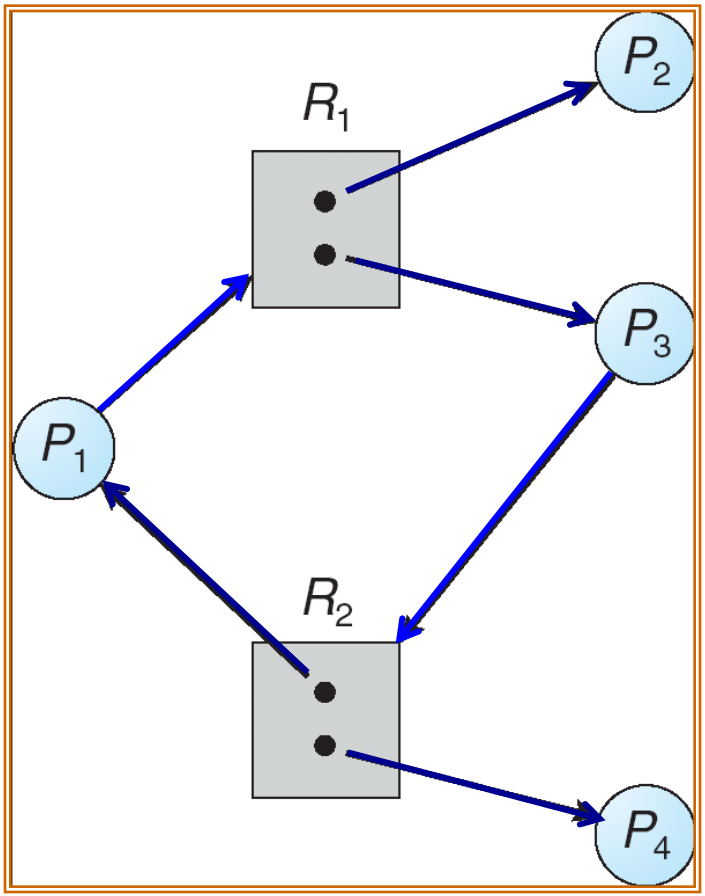
Esempio di grafo di allocazione delle risorse



Esempio di grafo di allocazione delle risorse con deadlock



Grafo di allocazione delle risorse con un ciclo, ma senza deadlock



?

Relazione tra cicli e deadlock

- Se il grafo non contiene cicli $\Rightarrow$ **no deadlock**
- Se il grafo contiene un ciclo:
  - Se c'è solo un'istanza per tipo di risorsa $\Rightarrow$ **deadlock**
  - Se ci sono più istanze per tipo di risorsa $\Rightarrow$ **possibilità di deadlock**

Metodi per gestire i deadlock

- Assicurare che il sistema non entri mai in uno stato di deadlock. (**Prevenzione**)
- Permettere al sistema di entrare in uno stato di deadlock, individuarlo, e quindi rimuoverlo. (**Detection**)
- **Ignorare** il problema, ipotizzando che le situazioni di stallo non si verificheranno mai nel sistema (approccio usato dalla maggior parte dei sistemi operativi, inclusi Windows e Linux)

Prevenire il deadlock (1/2)

Si può prevenire il deadlock assicurando che ***almeno una*** delle quattro condizioni necessarie al suo verificarsi non accada:

- **Mutua esclusione** – si dovrebbe impedire di chiedere la mutua esclusione su risorse condivisibili (**non praticabile!**)

- **Possesso e attesa** – si dovrebbe garantire che ogni volta che un processo richiede una risorsa, esso non possieda altre risorse:
  - Imporre ai processi di richiedere ed ottenere l'allocazione di tutte le risorse di cui necessitano prima di iniziare l'esecuzione, oppure permettere ai processi di richiedere risorse solo quando non ne possiedono già delle altre;
  - Basso utilizzo delle risorse; possibilità di starvation.

Prevenire il deadlock (2/2)

- **Impossibilità di Preemption** –
  - Se un processo che possiede alcune risorse e ne richiede altre che non possono essergli immediatamente assegnate, allora tutte le risorse che già possiede vengono rilasciate.
  - Le risorse rilasciate (prelazionate) sono aggiunte alla lista delle risorse per le quali il processo è in attesa.
  - Il processo verrà riavviato solo quando potrà riottenere le sue vecchie risorse insieme a quelle nuove che ha richiesto

- **Attesa circolare** – imporre un ordinamento totale all'insieme delle risorse, e richiedere che ogni processo richieda risorse secondo un ordine di numerazione crescente.

Evitare il deadlock

Gli algoritmi per evitare il deadlock richiedono che il sistema disponga *a priori* di alcune informazioni addizionali sulle modalità di richiesta delle risorse:

- Il modello più semplice richiede che ogni processo dichiari il *numero massimo* di risorse di cui potrebbe avere bisogno durante l'esecuzione

- **Date queste informazioni, si possono progettare algoritmi che esaminano dinamicamente lo *stato di allocazione delle risorse* per assicurare che non si verifichi mai un'attesa circolare**

- Lo *stato di allocazione delle risorse* è definito dal numero di risorse disponibili e assegnate e dalle richieste massime dei processi.

Stato sicuro

- Quando un processo richiede una risorsa disponibile, il sistema deve decidere se la sua immediata allocazione lascia il sistema in uno ***stato sicuro*** (*safe state*)
- Il sistema è in uno ***stato sicuro*** se esiste una possibile **sequenza**

  $<P_1, P_2, \ldots, P_n>$ che includa TUTTI i processi del sistema

tale che, per ogni P_i, le risorse che P_i può ancora richiedere possono essere **soddisfatte impiegando le risorse attualmente disponibili più le risorse possedute da tutti i P_j, con $j < i$**

- Quindi:
  - Se le risorse di cui necessita P_i non sono immediatamente disponibili, allora P_i attende fino a quando tutti i P_j hanno terminato
  - A quel punto, P_i può ottenere tutte le risorse di cui necessita, eseguire, restituire le risorse allocate, e terminare
  - Quando P_i termina, P_{i+1} può ottenere le risorse di cui necessita, e così via

Relazione tra stato sicuro e deadlock

- Se un sistema è in uno stato sicuro ⇒

  no deadlock

- Se un sistema è in uno stato non sicuro ⇒

  possibilità di deadlock

- Evitare il deadlock ⇒

  assicurare che un sistema non entri mai in uno stato non sicuro

Spazi degli stati sicuri, non sicuri e di deadlock