Lezione 15.2 [28/10/22]

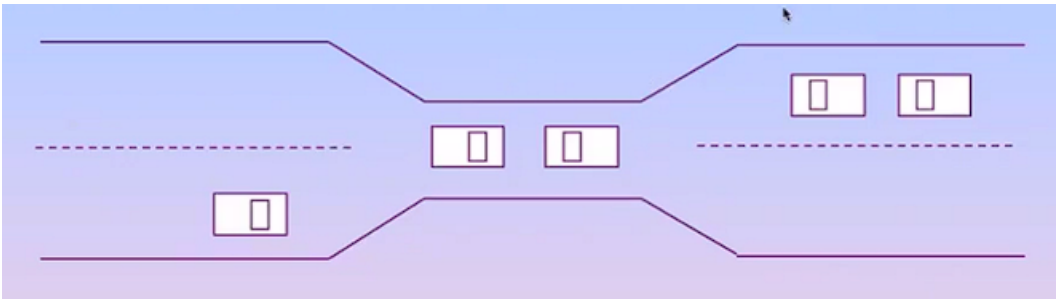
Capitolo 7: Stallo dei processi

- Modello del sistema
- Caratterizzazione delle situazioni di stallo
- Metodi per la gestione delle situazioni di stallo
- Prevenire le situazioni di stallo
- Evitare le situazioni di stallo
- Rilevamento delle situazioni di stallo
- Ripristino da situazioni di stallo

Stallo (Deadlock)

Un insieme di processi è in stallo se ciascun processo nell'insieme è in attesa di un evento che può essere causato solo da un altro dei processi nell'insieme

Esempio, attraversamento di un ponte:



In ogni istante, sul ponte possono transitare autoveicoli solo in una direzione

Ciascuna sezione del ponte può essere vista come una risorsa

Se si verifica una situazione di stallo, questa può essere risolta se un'auto torna indietro (rilascia la risorsa ed esegue un ritorno ad uno stato "sicuro" - **rollback**)

In caso di deadlock, può essere necessario che più auto debbano tornare indietro

È possibile si verifichi l'attesa indefinita (starvation)

Il problema del deadlock

In un ambiente multiprogrammato più processi possono entrare in competizione per cercare di ottenere risorse condivise

Se una risorsa non è correntemente disponibile, il processo richiedente entra in stato di attesa

Se le risorse sono trattenute da altri processi, a loro volta in stato di attesa, il processo potrebbe non poter più cambiare il proprio stato

Esempio:

- Nel sistema sono presenti solo due unità a nastri
- P_1 e P_2 hanno già avuto assegnate un'unità ciascuna
- Entrambi si bloccano e richiedono l'assegnazione di una seconda unità

Risorse riutilizzabili e risorse non riutilizzabili

Risorse riutilizzabili:

- Questo tipo di risorsa può essere usata in modo sicuro da un processo alla volta, ed essere riutilizzata in seguito da un altro processo
- Esempi di risorse riutilizzabili: processi, canali I/O, memoria centrale e secondaria, basi di dati, semafori, file, dispositivi, etc.

Risorse non riutilizzabili:

- Questo tipo di risorsa dopo essere stata creata, viene consumata (distrutta)
- Esempi di risorse non riutilizzabili: interrupt, segnali, messaggi, dati contenuti nel buffer I/O, etc.

Modello del sistema

Si assuma di avere risorse di tipo $R_1, R_2, ..., R_m$ (cicli di CPU, spazio di memoria, file, dispositivi di I/O)

Sono disponibili W_i istanze di ciascuna risorsa di tipo R_i

Nelle ordinarie condizioni di funzionamento un processo può servirsi di una risorsa solo se rispetta la seguente sequenza:

- **Richiesta:**
  - Il processo richiede la risorsa: se la richiesta non si può soddisfare immediatamente (ad es. la risorsa è attualmente in possesso di un altro processo) il processo richiedente deve attendere finché non possa acquisire tale risorsa
- **Utilizzo:**

    - Il processo può operare sulla risorsa (se, per esempio, la risorsa è una stampante, il processo può effettuare la stampa)
- **Rilascio:**
    - Il processo rilascia la risorsa

La richiesta e il rilascio di risorse avvengono tramite chiamate a sistema:

- request/release device, open/close file, allocate/free memory, o wait e signal sui semafori

Caratterizzazione dei deadlock

In una situazione di stallo i processi non terminano mai l'esecuzione, e le risorse del sistema vengono bloccate impedendo l'esecuzione di altri processi

Una situazione di deadlock può verificarsi solo se *(**potrebbe** esserci stallo se si verificano tutte le condizioni, ma non è certo)* valgono tutte e quattro le seguenti condizioni simultaneamente:

- **Mutua esclusione:**
    - Almeno una risorsa deve essere non condivisibile, vale a dire che è utilizzabile da un solo processo alla volta. Se un altro processo richiede tale risorsa, si deve ritardare il processo richiedente fino al rilascio della risorsa

- **Possesso e attesa:**
    - Un processo deve essere in possesso di almeno una risorsa e attendere di acquisire risorse già in possesso di altri processi
- **Assenza di prelazione:**
    - Le risorse non possono essere prelazionate, vale a dire che una risorsa può essere rilasciata dal processo che la possiede solo volontariamente, dopo aver terminato il proprio compito
- **Attesa circolare:**
    - Deve esistere un insieme { $P_0, P_1, ..., P_n$ } di processi, tale che P_0 attende una risorsa posseduta da P_1, P_1 attende una risorsa posseduta da P_2, …, P_{n-1} attende una risorsa posseduta da P_n e P_n attende una risorsa posseduta da P_0

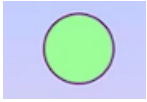
Grafo di assegnazione delle risorse

Le situazioni di stallo si possono descrivere con maggior precisione avvalendosi di una rappresentazione detta **grafo di assegnazione delle risorse**

Dato un processo P_i e una risorsa R_j, si chiama **arco di richiesta**, un arco orientato $P_i \rightarrow R_j$ (il processo P_i ha richiesto un'istanza del tipo di risorsa R_j), si chiama invece **arco di assegnazione**, un arco orientato $R_j \rightarrow P_i$ (un'istanza del tipo di risorsa R_j è assegnata al processo P_i)

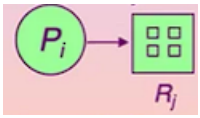
Un arco di richiesta è diretto soltanto verso il rettangolo del tipo di risorsa, mentre un arco di assegnazione deve designare anche uno dei puntini del rettangolo (l'istanza)

Processo:

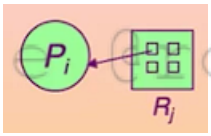


Tipo di risorsa con 4 istanze:

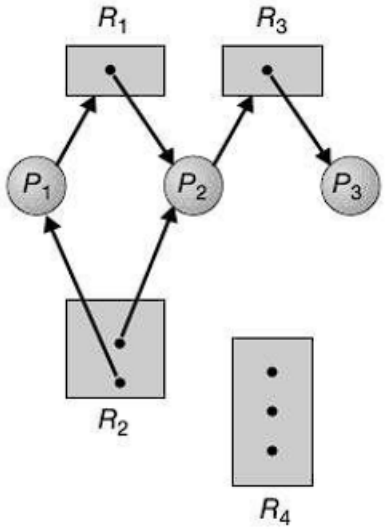
P_i richiede un'istanza di R_j



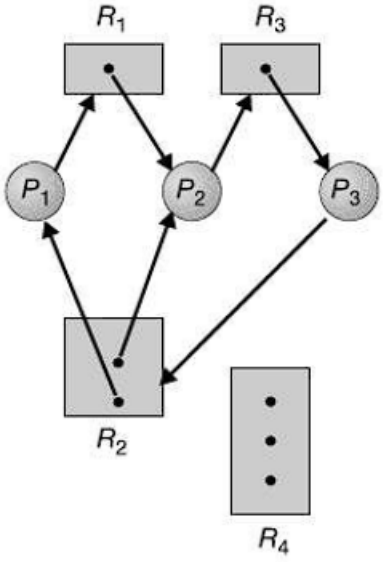
P_i possiede un'istanza di R_j



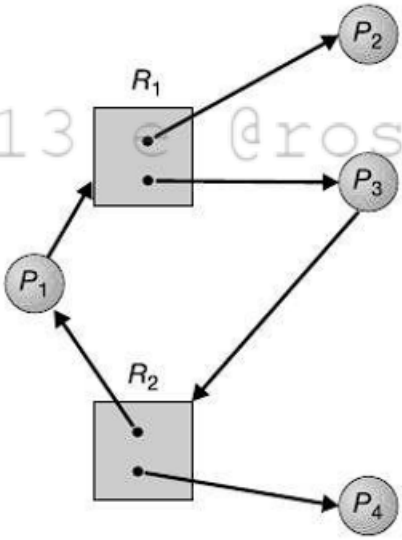
Grafo di assegnazione delle risorse:



Grafo di assegnazione delle risorse con deadlock:



Grafo di assegnazione delle risorse con un ciclo, ma senza deadlock:



Generalmente:

- Se il grafo non contiene cicli sicuramente non ci sono deadlock
- Se il grafo contiene un ciclo:
  - Se vi è una sola istanza per ogni tipo di risorsa, allora si ha un deadlock
  - Se si hanno più istanze per tipo di risorsa, allora il deadlock è possibile (ma non certo)

Metodi per la gestione dei deadlock

Essenzialmente, il problema delle situazioni di stallo si può affrontare in diversi modi:

- Assicurare che il sistema non entri mai in uno stato di deadlock
  - Prevenire i deadlock:
    - Evitare che contemporaneamente si verifichino mutua esclusione, possesso e attesa, impossibilità di prelazione e attesa circolare, la conseguenza ovviamente è anche un basso utilizzo di risorse e un throughput ridotto
  - Evitare i deadlock:
    - Evitare gli stati del sistema a partire dai quali si può evolvere verso un deadlock
- Permettere al sistema di entrare in uno stato di deadlock, quindi ripristinare il sistema
- Ignorare il problema e “fingere” che i deadlock non avvengano mai nel sistema; impiegato dalla maggior parte dei S.O., incluso UNIX

Per evitare le situazioni di stallo occorre che il S.O. abbia in anticipo informazioni riguardanti le risorse che verranno richieste e utilizzate da un processo; tramite questa informazioni il S.O. decide se soddisfare o meno una richiesta di risorsa (in caso non venisse soddisfatta, il processo attende)

Un S.O. può servirsi di un algoritmo per verificare la presenza di deadlock e, nel caso, utilizzare un secondo algoritmo per il ripristino del sistema

La presenza di situazioni di deadlock non rilevate causa un degrado delle prestazioni del sistema