LEZIONE 8

**Problema dello stallo**

* Un insieme di processi bloccati, in cui ciascun processo detiene una risorsa e attende di accedere a una risorsa in possesso di un altro processo.
* Esempio 1
* Il sistema dispone di 2 unità a nastri.
* P1 e P2 possiedono ciascuno una di queste unità e ciascuno richiede un’altra unità a nastri.
* Esempio 2 (stesso esempio della lezione precedente)

1. Se A e B, inizializzati a 1

P0 P1 wait(A); wait(B) wait(B); wait(A)

**Esempio dell’attraversamento del ponte**

* Traffico in una sola direzione
* Ciascuna sezione del ponte può essere considerata una risorsa
* Se si verifica uno stallo, può essere risolto se una macchina fa retromarcia: prelazione di risorse e ristabilimento di uno stato sicuro **(rollback)**

**ROLLBACK:** il rollback di un processo è molto complicata come cosa, in quanto significherebbe in qualche modo far arretrare il processo al suo stato precedente di computazione, sperando così che “molli” la risorsa usata per fare quel passo; quindi si sta eliminando in questo modo tutto ciò che è maturato da un certo punto in poi. Ciò potrebbe sbloccare la situazione perché qualche processo era magari in attesa di tale risorsa per proseguire nella sua elaborazione.

* Più macchine potrebbero dover fare retromarcia, in caso di stallo
* È possibile si verifichi un’attesa indefinita **(starvation)**

Modello del sistema Nel nostro sistema, stiamo parlando implicitamente di 2 unità:

1. Le risorse
2. Gli utilizzatori delle risorse

* Si può ipotizzare che ogni sistema abbia tipi di risorse R1,R2,…,Rm (cicli di CPU, spazio di memoria, dispositivi di I/O)
* Ciascun tipo di risorsa Ri ha Wi istanze (con Ri si intende una classe di risorse che è composta da W istanze. Es. si hanno 5 stampanti quando si effettua una richiesta il SO decide quale assegnare a chi ha effettuato la richiesta, se questa ovviamente è disponibile)
* Ciascun processo utilizza una risorsa nella seguente maniera:

1. Richiesta
2. Uso
3. Rilascio

È molto importante stabilire il punto d’intervento, cioè dove intervenire per tentare di evitare i deadlock. Si potrebbe intervenire nella fase di richiesta, dato che è una fase preventiva; oppure nella fase d’uso, se il sistema accetta prelazioni (cioè si può bloccare il processo in esecuzione)

Caratterizzazione delle situazioni di stallo Si può avere una situazione di stallo solo se si verificano contemporaneamente le seguenti 4 condizioni:

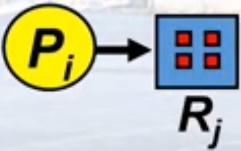
1. **Mutua esclusione:** solo un processo alla volta può utilizzare una risorsa
2. **Possesso attesa:** un processo in possesso di almeno una risorsa attende di acquisire risorse già in possesso di altri processi
3. **Impossibilità di prelazione:** una risorsa può essere rilasciata dal processo che la possiede solo volontariamente, dopo aver terminato il proprio compito
4. **Attesa circolare:** deve esistere in insieme {P0,P1,…,Pn} di processi tale che P0 è in possesso di una risorsa, e attende una risorsa in possesso da P1, P1 attende una risorsa posseduta da P2,…, Pn-1 attende una risorsa posseduta da Pn, Pn attende una risorsa da P0.

Grafo di assegnazione delle risorse Uno dei metodi per prevenire lo stallo è l’uso di un grafo. Quindi bisogna trovare un isomorfismo tra le entità che sono presenti in un sistema con un grafo. Si sa che in un sistema ci sono processi, risorse, processi che richiedono risorse e risorse assegnate a processi (sono i 4 elementi da identificare nella piattaforma; due tipi di vertici/nodi, e due tipi di archi) **Un insieme si vertici V e di archi E** **Tipi di vertici:**

1. P= {P1, P2,…,Pn}=tutti i processi
2. R={R1,R2,…,Rm}=tutti i tipi di risorse (ogni R indica una classe di risorse)

**Tipi di archi:**

1. Arco di richiesta: arco orientato P1 -> Rj
2. Arco di assegnazione: arco orientato Rj -> Pi

I processi si indicano così: I tipi di risorse si indicano così: -> classe di risorse che ha 4 istanze

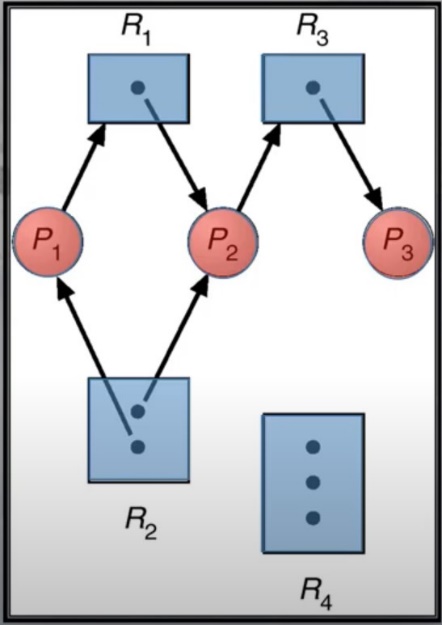
Pi richiede un’istanza del tipo di risorsa Rj Pi punta alla classe

Immagine che contiene orologio, disegnando, metro, segnale

Descrizione generata automaticamentePi possiede un’istanza del tipo di risorsa Rj se la richiesta è stata approvata, ovvero la risorsa è disponibile, il SO ne concede una al processo richiedente (infatti la freccia parte dal quadratino). Quindi si dice che Pi possiede un’istanza della classe Rj **Esempio Grafo di assegnazione delle risorse**

**Immagine che contiene orologio, oggetto, diverso

Descrizione generata automaticamente**

**Caso in cui si crea un ciclo**

**Spiegazione** P3 chiede qualcosa a R2, ma questo ha già tutte le risorse assegnate, di conseguenza il processo P dovrà aspettare, ma la domanda è: **questa attesa è temporanea oppure è un blocco?** In questo caso si dice che il grafo di assegnazione, coi permette di trovare una situazione di stallo, dato che se partendo da un nodo, e attraversando un sottoinsieme di archi e di nodi, ci porta al nodo di partenza.

In questo caso ci sono 2 cicli:

1. P1 -> R1 -> P2 -> R3 -> P3 -> R2 -> P1
2. P2 -> R3 -> P3 -> R2 -> P2 Questi due cicli sono blocchi, quindi stallo perché è una situazione di possesso-attesa.

**Quindi il ciclo è un sintomo di allerta per lo stallo; però non è detto che un ciclo conduca per forza ad uno stallo infatti:**

**Immagine che contiene orologio, oggetto, disegnando

Descrizione generata automaticamenteNOTA:** P1 -> R1 -> P3 -> R2 è un ciclo interno che non conduce ad uno stallo perché P4 come P2 usano una risorsa, ma non attendono nulla.

Osservazioni

* **Se il grafo NON contiene cicli**
* Non si verificano situazioni di stallo
* **Se il grafo contiene un ciclo**
* Se c’è solo un’istanza per tipo di risorsa, allora si verifica una situazione di stallo
* Se vi sono più istanze per tipo di risorsa, allora c’è la **possibilità** che si verifichi una situazione di stallo

Metodi per la gestione delle situazioni di stallo

* Assicurare che il sistema non entri mai in stallo (prevenzione)
* Consentire al sistema di entrare in stallo, individuarlo e quindi eseguire il ripristino (cura)
* Ignorare del tutto il problema “fingendo” che le situazioni di stallo non possano mai verificarsi nel sistema; è la soluzione usata nella maggior parte dei SO, compreso UNIX (se il deadlock si verifica ogni 50 anni, allora si applica l’algoritmo dello struzzo(far finta di niente))

Prevenire le situazioni di stallo

**Si può prevenire il verificarsi di uno stallo assicurando che almeno una delle 4 condizioni necessarie non si verifichi.**

* **Mutua esclusione:** non richiesta per le risorse condivisibili; deve invece **valere per le risorse non condivisibili.**
* **Possesso attesa:**
* Occorre garantire che ogni volta che un processo richiede una risorsa, non ne possegga altre
* Ogni processo, prima di iniziare la propria esecuzione, deve richiedere tutte le risorse che gli servono, e queste gli devono essere assegnate; oppure occorre permettere a un processo di richiedere risorse solo se non ne possiede

**Significa che un processo prende le risorse solo se sono tutte disponibili.**

* Basso utilizzo delle risorse; possibile attesa indefinita

**Significa che si può correre in un sottoutilizzo delle risorse**

* **Impossibilità di prelazione:**
* Se un processo che possiede una o più risorse ne richiede un’altra che non gli si può assegnare immediatamente, allora si esercita la prelazione su tutte le risorse attualmente in suo possesso
* Le risorse si aggiungono alla lista delle risorse che il processo sta attendendo
* Il processo viene nuovamente avviato solo quando può ottenere sia le vecchie risorse sia quelle che sta richiedendo.

**Bisogna comunque stare attenti in quanto anche se un processo abbastanza complicato viene prelazionato, questo comunque perde una parte di calcolo, di conseguenza bisogna andarci cauti.**

* **Attesa circolare:**
* Impone un ordinamento totale all’insieme di tutti i tipi di risorse e un ordine crescente di numerazione per le risorse richieste da ciascun processo.

**Anche qui si può correre in un sottoutilizzo delle risorse**

**RICORDA:** ognuna di queste soluzioni non si può mai garantire al 100%, solo dove si può fare qualche miglioramento per far si che si abbassino le probabilità di stallo

Evitare le situazioni di stallo Un metodo alternativo per evitare le situazioni di stallo consiste nel richiedere ulteriori informazioni sui modi di richiesta delle risorse.

* Il modello più utile e più semplice richiede che **ciascun processo dichiari il numero massimo delle risorse di ciascun tipo di cui necessita**
* L’algoritmo per evitare lo stallo esamina dinamicamente lo stato di assegnazione delle risorse per garantire che non possa esistere una condizione di attesa circolare
* **Lo stato di assegnazione delle risorse è definito dal numero di risorse disponibili e assegnate, e dalle richieste massime dei processi**

Stato sicuro

* Uno stato di dice sicuro se il sistema è in grado di **assegnare risorse a ciascun processo (fino al massimo) in un certo ordine** e impedire il verificarsi di uno stallo
* Un sistema si trova in uno stato sicuro solo se esiste una sequenza sicura
* **La sequenza <P1, P2,…, Pn> è sicura se per ogni Pi, le richieste che Pi può ancora fare si possono soddisfare impiegando le risorse attualmente disponibili ed in n+ le risorse possedute da tutti i Pj, con j<i**
* Se le risorse necessarie al processo Pi non sono immediatamente disponibili, allora Pi può attendere che tutti i Pj abbiano finito
* A quel punto, Pi può ottenere tutte le risorse di cui ha bisogno, completare il compito assegnato, restituire le risorse assegnate, e terminare
* Quando Pi termina, Pi+1 può ottenere le risorse richieste, e così via

Osservazioni

* Se un sistema è in uno **stato sicuro => non si verificano situazioni di stallo**
* Se un sistema è in uno **stato non sicuro => c’è possibilità di stallo, ma non è detto che avvenga sicuramente**
* Evitare lo stallo => assicurare che il sistema non si trovi mai in uno stato non sicuro **Graficamente lo potremo vedere in questo modo**

**NON SICURO**

**SICURO**

**STALLO**

Algoritmo con grafo di assegnazione delle risorse

* **ARCO DI RECLAMO:** stabilisce una condizione, ovvero, il processo Pi potrebbe richiedere una risorsa Rj (ma non è detto che lo farà). È rappresentato con una linea tratteggiata
* Quando la richiesta per la risorsa del processo diventa effettiva allora l’arco di reclamo diventa un **arco di richiesta**
* Quando un processo rilascia una risorsa, l’arco di assegnazione ridiventa un arco di reclamo
* **Immagine che contiene oggetto, orologio, fotografia, diverso

  Descrizione generata automaticamenteLe risorse devono essere reclamate a priori nel sistema**

**Spiegazione** Sia P1 che P2 vogliono la risorsa in R2; ci sono due possibilità: 1. **Se prima P1** **accede a R2 allora siamo in** **uno stato sicuro** in quanto non si crea circolarità. 2. **Se P2 accede per primo a R2 invece, siamo in uno stato insicuro**, in quanto si crea circolarità che potrebbe diventare uno stallo

Algoritmo del banchiere A differenza dell’algoritmo del grafo di assegnazione che funziona se si fa l’ipotesi che per ogni classe di risorse ci sia una sola istanza. Con l’algoritmo del banchiere invece ha questi vantaggi:

* Istanze multiple (cioè una classe di risorse può avere più istanze)
* Ciascun processo deve dichiarare a priori il numero massimo delle istanze di ciascun tipo di risorsa di cui necessita (cioè dichiarare tutte le richieste che potrebbe fare)
* Quando un processo richiede una risorsa può dover attendere
* Quando un processo ottiene tutte le sue risorse, deve restituire entro un intervallo di tempo definito (così si evita starvation)

Struttura dati per l’algoritmo del banchiere Sia n il numero dei processi nel sistema e m il numero dei tipi di risorsa

* **Disponibili:** un vettore di lunghezza m indica il numero delle istanze disponibili per ciascun tipo di risorsa. **Se disponibili[j] = k, significa che sono disponibili k istanze del tipo di risorsa Rj**
* **Massimo:** una matrice n x m definisce la richiesta massima di ciascun processo. **Massimo[i,j] = k, significa che il processo Pi può richiedere al più k istanze del tipo di risorsa Rj**
* **Assegnate:** una matrice n x m definisce il numero delle istanze di ciascun tipo di risorsa attualmente assegnante a ciascun tipo di processo. **Assegnate[i,j] = k, significa** **che al processo Pi** **sono assegnate k istanze del tipo di risorsa Rj**
* **Necessità:** una m,atrice n x m indica la necessità residua di risorse relative a ogni processo. **Necessità[i,j] = k, significa che il processo Pi può aver bisogno di altre istanze di RJ per completare il suo compito RICORDA BENE: NECESSITA’[i,j] = MASSIMO[i,j] – ASSEGNATE[i,j]**

Algoritmo di verifica della sicurezza

1. Siano Lavoro e Fine vettore di lunghezza rispettivamente m e n. Inizializza: **Lavoro = Disponibili Fine[i] = falso per i-1,3,…,n**
2. Cerca un indice i tale che: **(a) Fine[i] = falso (b) Necessità ≤ Lavoro se tale i non esiste esegui il passo 4**
3. Lavoro := Lavoro + Assegnate **Fine[i] := vero torna al passo 2**
4. **Se Fine[i] == vero per ogni i, allora il sistema è in uno stato sicuro**

Svantaggi per l’algoritmo del banchiere l’applicazione dell’algoritmo del banchiere presuppone una conoscenza completa del sistema… Rilevamento delle situazioni di stallo Se un sistema non si avvale di un algoritmo per prevenire o per evitare le situazioni di stallo, è possibile che una situazione di stallo si verifichi effettivamente. In tal caso il sistema deve fornire i seguenti algoritmi:

* Un algoritmo che esamini lo stato del sistema per stabilire se si è verificato uno stallo
* Un algoritmo che ripristini il sistema della condizione di stallo Istanza singola di ciascun tipo di risorsa
* Grafo d’attesa
* I nodi sono processi
* Pi -> Pj se Pi è in attesa di Pj
* Per individuare le situazioni di stallo il sistema deve conservare il grafo d’attesa e invocare periodicamente un algoritmo che cerchi un ciclo all’interno del grafo
* L’algoritmo per il rilevamento di un ciclo all’interno di un grafo richiede un numero di operazioni dell’ordine di , dove con n si intende il numero dei vertici del grafo **Grafo di assegnazione delle risorse e grafo d’attesa**

Immagine che contiene fotografia, sedendo, tavolo, computer

Descrizione generata automaticamentePartendo dalla figura (a), togliendi i rettangoli si ottiene la figura (b). Se poi in seguito all’interno della figura (b) si trova un ciclo c’è qualcosa che non va.

**RICORDA:** trovare un ciclo all’interno di un grafo è molto dispendioso, ovvero,

Più istanze di ciascun tipo di risorsa

* **Disponibili:** vettore di lunghezza m che indica il numero delle istanze disponibili per ciascun tipo di risorsa
* **Assegnate:** Matrice n x m che definisce il numero delle istanze di ciascun tipo di risorse correntemente assegnate a ciascun processo
* **Richieste:** Matrice n x m che indica la richiesta attuale di ciascun processo. Se Richieste[ij] = k, significa che il processo Pi sta richiedendo altre risorse k del tipo di risorsa Rj Algoritmo di rilevamento

1. Siano Lavoro e Fine due vettori di lunghezza rispettivamente n e n. inizializza**: (a) Lavoro = Disponibili (b) per i = 1,2,…,n, se Assegnate ≠ 0, allora Fine[i] = falso; altrimenti, Fine[i] = vero**
2. Cerca un indice i tale che: **(a) Fine[i] == falso (b) Richieste ≤ Lavoro se tale i non esiste, esegui passo 4**
3. Lavoro := Lavoro + Assegnate **Fine[i] = vero torna al passo 2**
4. **Se Fine[i] == falso, per qualche i, 1 ≤ i ≤ n, allora il sistema è in stallo. Inoltre, se Fine[i]==falso, allora il processo Pi è in stallo**

**Tale algoritmo richiede un numero di operazioni dell’ordine di m x per controllare se il sistema è in stallo**

Uso dell’algoritmo di rilevamento

* Per sapere quando è necessario ricorrere all’algoritmo di rilevamento, occorre considerare i seguenti fattori:
* **Frequenza** (presunta) con la quale si verifica uno stallo
* **Numero dei processi** che sarebbero influenzati da tale stallo
* Non è conveniente richiedere l’algoritmo di rilevamento in momenti arbitrari, poiché nel grafo delle risorse possono coesistere molti cicli e, normalmente, non si può dire quale fra i tanti processi in stallo abbia “causato” lo stallo

Ripristino delle situazioni di stallo: terminazione di processi

* Terminazione di tutti i processi in stallo
* Terminazione di un processo alla volta fino all’eliminazione del ciclo di stallo
* In quale ordine effettuare la terminazione?
* Priorità dei processi
* Tempo trascorso dalla computazione e tempo ancora necessario per completare i compiti assegnati ai processi
* Quantità e tipo di risorse impiegate dai processi
* Quantità di ulteriori risorse di cui i processi hanno ancora bisogno per completare i propri compiti
* Numero di processi che si devono terminare
* Tipo di processi: interattivi o a lotti

Ripristino delle situazioni di stallo: prelazione di stallo

* **Selezione di una vittima:** minimizzare i costi
* **Rollback:** ristabilimento di un precedente stato sicuro, dal quale il processo può essere riavviato
* **Attesa indefinita (starvation):** può accadere che si scelga sempre lo stesso processo come vittima

Approccio combinato per la gestione dello stallo

* La combinazione dei tre approcci di base Consente l’uso dell’applicazione ottimale per ciascuna delle risorse del sistema:
* Prevenire
* Evitare
* Rilevare
* Partizione delle risorse in classi ordinate gerarchicamente
* Uso della tecnica più appropriata per la gestione dello stallo all’interno di ciascuna classe