**Deadlock**

Un deadlock è una situazione che si verifica quando un processo richiede una risorsa ma questa è non disponibile il processo richiedente passa nello stato d’attesa. Se le richieste sono trattenute da altri processi a loro volta nello stato d’attesa, il processo potrebbe non cambiare mai il suo stato e situazioni di questo tipo sono dette deadlock.

Si può avere una situazione deadlock solo se si verificano contemporaneamente quattro condizioni:

1. Mutua esclusione una risorsa non è condivisibile, ovvero è utilizzabile da un solo processo alla volta.
2. Possesso e attesa un processo che è in possesso almeno di una risorsa attende di acquisire delle risorse in possesso ad altri processi.
3. Impossibilità di prelazione non esiste il diritto di prelazione sulle risorse, un processo rilascerà le risorse solo quando avrà terminato il suo compito.
4. Attesa circolare deve esistere un insieme di processi ad esempio da P0 a Pn dove P0 attende una risorsa posseduta da P1, P1 attende una risorsa posseduta da P2, Pn-1 attende una risorsa posseduta da Pn, Pn attende una risorsa posseduta da P0.

Prevenire le situazioni di stallo significa far uso di metodi atti ad assicurare che non si verifichi una delle condizioni necessarie.

Quindi affinché si abbia un deadlock si devono verificare 4 condizioni necessarie, perciò si può prevenire un deadlock impedendo che una della 4 condizioni posso capitare.

**Mutua esclusione** Non si può prevenire un deadlock negando la condizione di mutua esclusione poiché alcune risorse sono intrinsecamente non condivisibili.

**Possesso e attesa** Per prevenire una condizione di possesso e attesa occorre garantire che un processo che richiede una risorsa non ne possegga altre.

Si può usare un protocollo che ponga la condizione che ogni processo, prima di iniziare la sua esecuzione richieda tutte le risorse che gli servono e che esse gli siano assegnate. Questa condizione si può realizzare imponendo che le chiamate di sistema che richiedono una risorsa debbano avere la precedenza su tutte le altre.

Un protocollo alternativo prevede che un processo possa richiedere delle risorse solo se non ne possiede. Un processo può richiedere delle risorse e adoperarle, ma prima di richiederne altre deve rilasciare tutte quelle in suo possesso.

Entrambi i due protocolli presentano due svantaggi principali. Innanzitutto l’utilizzo delle risorse può risultare poco efficiente, poiché molte risorse possono essere assegnate ma non utilizzate per un lungo periodo di tempo.

Il secondo svantaggio è dovuto al fatto che si possono verificare situazioni di attesa indefinita. Un processo che richiede più risorse può trovarsi nella condizione di attenderne indefinitamente la disponibilità poiché almeno una delle risorse di cui necessità è sempre assegnata a qualche altro processo.

**Impossibilità di prelazione** Un protocollo che previene la terza condizione prevede che un processo in possesso di risorse se effettua una richiesta per una risorsa ma questa risorsa è attualmente non disponibile allora si ha il diritto di prelazione sulle risorse posseduta da quel processo. Si ha un rilascio implicito delle risorse, che 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.

Questo protocollo è adatto a risorse il cui stato si può salvare e recuperare facilmente in un secondo tempo come i registri delle CPU o lo spazio di memoria, mentre non si può applicare a risorse come le stampanti e le unità a nastri.

**Attesa circolare** La cosa più decente che si può fare dal punto di vista della prevenzione è agire sulla attesa circolare. Un modo per assicurare che tale condizione non si verifichi consiste nell’imporre un riordino delle risorse assegnando loro un numero in ordine crescente.

Per evitare il deadlock un processo può richiedere solo risorse superiori a quella posseduta.

**Evitare i deadlock**

I metodi per evitare i deadlock prevedono che un processo dichiari sin dall’inizio le risorse di cui avrà bisogno quindi non solo le risorse di cui ha bisogno immediatamente ma anche di quelle future.

Uno stato si dice sicuro se il sistema è in grado in assegnare risorse a ciascun processo in un certo ordine impedendo il verificarsi di deadlock.

Quindi uno stato sicuro è un stato in cui qualsiasi cosa succede e qualsiasi richiesta mi pervenga io riesco a far terminare i processi evitando i deadlock.

Inizialmente il sistema si trova in uno stato sicuro, ogni volta che un processo richiede una risorsa, il sistema deve stabilire se la risorsa è attualmente disponibile oppure il processo debba attendere.

Si soddisfa la richiesta solo se l’assegnazione lascia il sistema in uno stato sicuro.

Abbiamo due algoritmi che permettono di evitare i deadlock

Per risorse a singole istanze si utilizza **un grafo di allocazione delle risorse**. Nel grafo oltre alle richieste e alle assegnazioni sono presenti anche le richieste future. Se la conversione dell’arco richiesta in un arco assegnazione non porta alla creazione di un ciclo allora la risorsa può essergli assegnata, altrimenti no perché il sistema si troverebbe in uno stato insicuro, quindi potenzialmente incline a deadlock.

Per risorse a istanze multiple si utilizza **l’algoritmo del banchiere**.

L’algoritmo del banchiere prevede che quando si presenti un nuovo processo questo dichiari il numero massimo dell’istanze delle risorse di cui necessita, questo numero non può superare il numero totale delle risorse del sistema. Quando un processo richiede un gruppo di risorse, si deve stabilire se l’assegnazione di queste risorse lasci il sistema in uno stato sicuro, se si rispetta tale condizione, si assegnano le risorse altrimenti il processo deve attendere che qualche altro processo ne rilasci un numero sufficiente.