



Corso di Laurea Triennale in Informatica
Università Degli Studi della Basilicata

Sistemi Operativi

Docente:
ugo.erra@unibas.it

7° Lezione

Stallo dei processi

Sommario della lezione



- Definizione dello stallo
- Condizioni necessarie
- Grafo di assegnazione delle risorse
- Metodi per la gestione delle situazioni di stallo
 - ▣ Prevenire lo stallo
 - ▣ Evitare lo stallo
- Rilevamento delle situazioni di stallo
 - ▣ Ripristino dallo stallo

Modello del sistema - 1

- Uno scopo di un Sistema Operativo è la gestione delle risorse di sistema da assegnare tra i processi in esecuzione
 - ▣ Cicli di CPU, spazio di memoria, file e dispositivi di I/O, etc...
- Per ogni risorsa disponibile possiamo definire il numero di istanze dello stesso tipo
 - ▣ Ad esempio una stampante potrebbe avere 5 istanze per indicare che sono presenti nel sistema 5 stampanti
- Assegnare una risorsa significa assegnare una istanza di quella risorsa al processo che ne richiede l'uso

Modello del sistema - 2

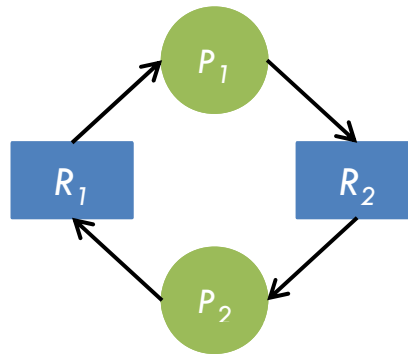


- In un Sistema Operativo un processo richiede una risorsa attraverso la seguente sequenza:
 1. Richiesta
 2. Uso
 3. Rilascio

- Se il Sistema Operativo non può soddisfare la richiesta il processo viene accodato ad altri processi che ne hanno fatto richiesta

Il problema del deadlock

- In questo scenario un gruppo di processi entra in **stallo** quando attende il rilascio di una risorsa che si trova nella situazione di attesa per il rilascio di un'altra risorsa



- In passato il problema dello stallo era evitato a causa della sua inefficiente gestione ma con l'aumento del numero di thread lo scenario potrebbe diventare critico

Caratterizzazione delle situazioni di stallo

- Il Sistema Operativo finisce in uno stallo solo se si verificano le seguenti condizioni:
 - **Mutua esclusione**
 - Esiste una risorsa non condivisibile utilizzabile da un solo processo. Se un processo richiede questa risorsa deve accodarsi fino al suo rilascio
 - **Possesso e attesa**
 - Un processo in possesso di almeno una risorsa cerca di acquisire una risorsa posseduta già da un altro processo
 - **Impossibilità di prelazione**
 - Una risorsa può essere rilasciata volontariamente solo dal processo che ne è in possesso e dopo aver completato il suo compito, cioè non si possono rubare risorse già in uso da altri processi
 - **Attesa circolare**
 - Esiste un insieme di processi in attesa $\{P_1, P_2, \dots, P_n\}$ tale che il processo P_1 attende una risorse posseduta da P_2 , P_2 attende una risorsa posseduta da P_3, \dots, P_n attende una risorsa posseduta da P_1

Grafo di assegnazione delle risorse - 1

- Le situazioni di stallo si possono descrivere utilizzando il **grafo di rappresentazione delle risorse** $G=(V, E)$ in cui:
 - ▣ L'insieme dei vertici V rappresenta l'insieme dei processi $P = \{P_1, P_2, \dots, P_n\}$ e l'insieme delle risorse $R = \{R_1, R_2, \dots, R_m\}$
 - ▣ L'insieme degli archi E rappresenta:
 - Un **arco di richiesta** $P_i \rightarrow R_j$ se il processo P_i ha richiesto l'uso della risorsa R_j
 - Un **arco di assegnazione** $R_j \rightarrow P_i$ se la risorsa R_j è assegnata al processo P_i

Grafo di assegnazione delle risorse - 2

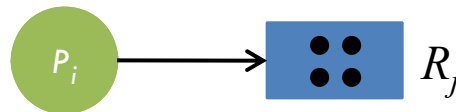
- Un processo



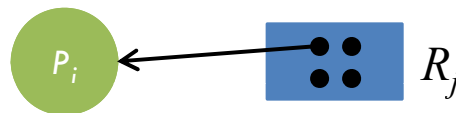
- Una risorsa con 4 istanze



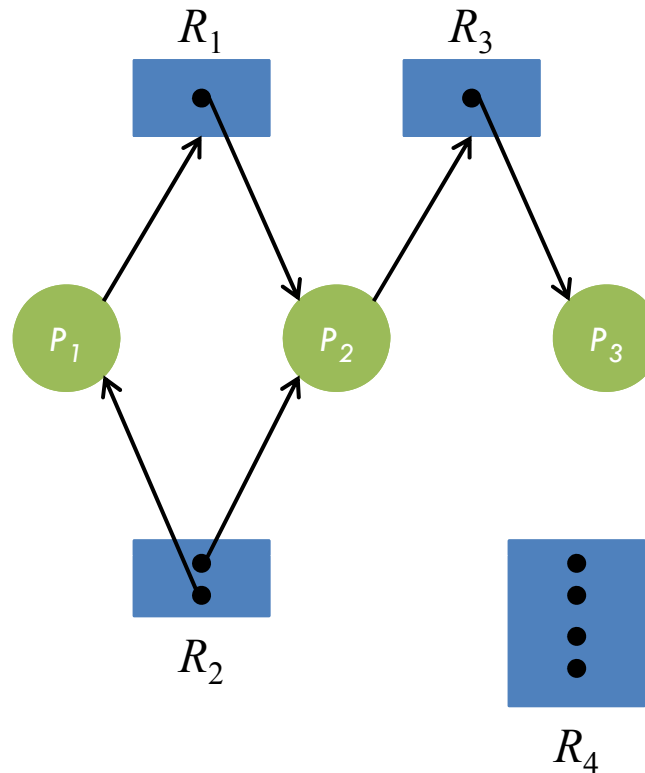
- Il processo P_i chiede una istanza di R_j



- Il processo P_i possiede una istanza della risorsa R_j

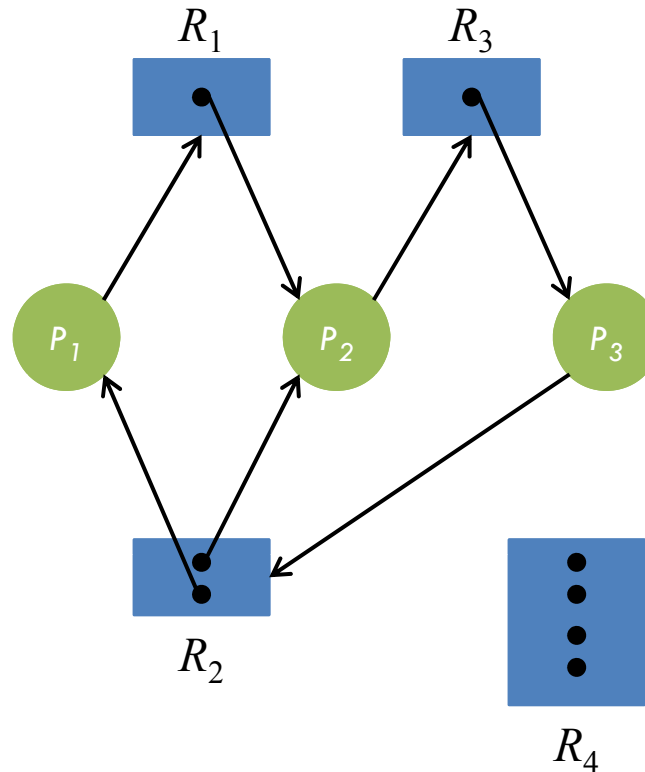


Esempio allocazione delle risorse - 1



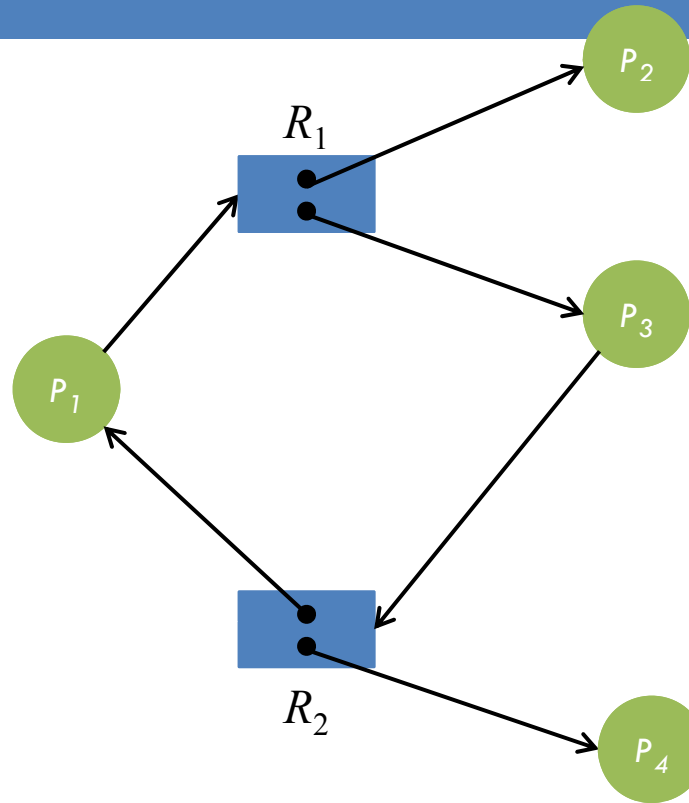
- Se il grafo non contiene cicli nessun processo del sistema subisce uno stallo

Esempio allocazione delle risorse - 2



- Il grafo contiene due cicli ed i processi P_1, P_2, P_3 sono in una situazione di stallo

Esempio allocazione delle risorse - 3



- Il grafo contiene un ciclo ma questo non implica la presenza di uno stallo
 - ▣ Se il processo P_4 libera la risorsa R_2 questa viene assegnata al processo P_3 che rompe il ciclo

Conclusioni sul grafo di assegnazione delle risorse

- Se il grafo non contiene cicli \Rightarrow non c'è deadlock
- Se il grafo contiene un ciclo allora:
 - ▣ Se esiste una sola istanza per risorsa \Rightarrow c'è un deadlock
 - ▣ Se ci sono diverse istanze per risorsa \Rightarrow non c'è necessariamente un deadlock
- In conclusione:
 1. L'assenza di cicli nel grafo di assegnazione delle risorse implica l'assenza di situazioni di stallo
 2. La presenza di cicli nel grafo di assegnazione delle risorse non è condizione sufficiente per implicare la presenza di uno stallo nel sistema

Metodi per la gestione dei deadlock

- Le situazioni di stallo si possono affrontare nei seguenti modi:
 - ▣ Assicurarci che il sistema non entri *mai* in una situazione di stallo
 - ▣ Permettere al sistema di entrare in una situazione di stallo e quindi eseguire un ripristino
 - ▣ Ignorare il problema fingendo che il sistema non entrerà mai in una situazione di stallo (Windows, UNIX)

Metodi per non causare situazioni di stallo

- Per assicurarsi che il sistema non entri mai in una situazione di stallo il sistema può:
 - **Prevenire le situazioni di stallo**
 - Il Sistema Operativo si incarica di verificare che almeno una delle condizioni necessarie non si verifichi
 - **Evitare le situazioni di stallo**
 - Il Sistema Operativo deve conoscere in anticipo le risorse di cui avrà bisogno il processo in modo da determinare se il sistema rischia di entrare in una situazione di stallo

Prevenire la mutua esclusione

- La mutua esclusione deve valere per le risorse non condivisibili ad esempio una stampante non può essere condivisa tra più processi, ma...
- Un processo non deve (o dovrebbe) mai attenersi a una risorsa condivisibile
- Alcune risorse sono per natura non condivisibili quindi è difficile prevenire lo stallo negando la condizione di mutua esclusione
 - ▣ Lo Spooling consiste di un processo demone (*daemon*) e una directory di spooling
 - ▣ Il processo sarà l'unico ad avere accesso alla risorsa. Tutti i processi si limitano ad inserire l'output nella directory
 - ▣ Non è utilizzabile con tutti i tipi di risorse, ma è ottima per ad esempio stampanti, e-mail, ftp, etc...

Prevenire il possesso e attesa

- Bisogna garantire che un processo che richiede una risorsa non ne possieda altre
- Due sono i possibili protocolli che si possono usare per evitare il possesso e attesa:
 1. Un processo prima di iniziare la sua esecuzione dichiara quali saranno le risorse da utilizzare
 2. Un processo può utilizzare delle risorse solo se ha rilasciato le risorse precedenti
- In ogni caso sono possibili situazioni di starvation in cui un processo che richiede risorse molto utilizzate non gli verranno mai assegnate

Prevenire l'impossibilità di prelazione

- La condizione richiede che non sia possibile avere la prelazione su risorse già assegnate
- Sicuramente non è possibile prelaionare una risorsa già assegnata ad un processo
 - ▣ Una stampante viene assegnata ad un altro processo mentre è in corso una stampa
- Si può verificare che se un processo richiede una nuova risorsa e non può essere concessa allora tutte le risorse già in suo possesso vengono prelaionate
 - ▣ Tutte le risorse vengono implicitamente rilasciate e si aggiungono alla lista delle risorse che il processo richiede
- Fino a quando il processo non ottiene sia le vecchie risorse che quelle nuove non viene avviato

Prevenire l'attesa circolare

- Per evitare che questa condizione non si verifichi mai è imporre un ordine totale a tutte le risorse e richiedere che ogni risorsa richiesta dal processo venga richiesta in ordine crescente

- Supponiamo di avere un insieme di risorse

$$\{R_1, R_2, \dots, R_m\}$$

- Per ogni risorsa associamo una funzione

$$f: R \rightarrow N$$

- Un processo in possesso di una risorsa R_i può richiedere una nuova risorsa R_j se e solo se $f(R_j) > f(R_i)$
- Utilizzando questa condizione non è possibile la creazione di un'attesa circolare

Evitare le situazioni di stallo

- Per evitare le situazioni di stallo il sistema cerca di determinare se l'assegnazione di una risorsa può provocare una situazione di stallo
 - ▣ Lo svantaggio di questa soluzione può comportare una ridotta produttività delle risorse
- Un algoritmo per evitare lo stallo richiede che ogni processo dichiari il *numero massimo* di risorse che intende utilizzare
- Lo *stato* di assegnazione delle risorse è definito come il numero di:
 - ▣ Risorse disponibili
 - ▣ Richieste massime dei processi
 - ▣ Risorse assegnate
- Un algoritmo per **evitare lo stallo** esamina lo stato di assegnazione per garantire che non ci sia una condizione di attesa circolare

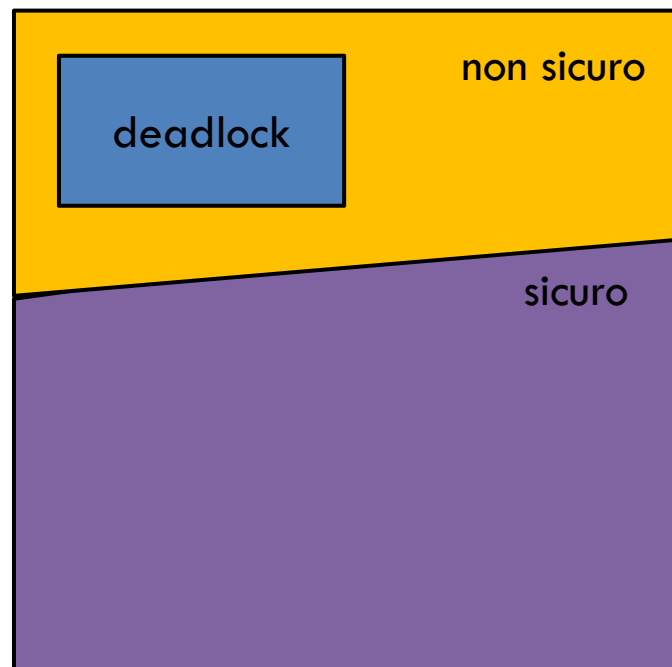
Stato sicuro



- Quando un processo richiede una risorsa disponibile, il sistema deve decidere se l'assegnazione di quella risorsa lascia il sistema in uno **stato sicuro**

Spazio degli stati

- Le considerazioni da fare sugli stati sono:
 - ▣ Se il sistema è in uno stato sicuro \Rightarrow non ci sono deadlock
 - ▣ Se il sistema è in uno stato non sicuro \Rightarrow c'è possibilità di deadlock
- Evitare che il sistema entri in uno stato non sicuro



Sequenza sicura

- Una sequenza di processi $\langle P_1, P_2, \dots, P_n \rangle$ è definita come una **sequenza sicura** se per ogni richiesta di P_i il sistema è in grado di soddisfare la richiesta:
 1. Impiegando le risorse disponibili
 2. Impiegano le risorse possedute da tutti i processi P_j con $j < i$
- Il sistema si trova in uno stato sicuro solo se esiste una sequenza sicura

Esempio

- Consideriamo un sistema con:
 - ▣ 3 processi P_0, P_1, P_2
 - ▣ 12 unità a nastri
- Per ogni processo il numero di richieste massime sono:
 - ▣ P_0 può richiedere al max 10 unità
 - ▣ P_1 può richiedere al max 4 unità
 - ▣ P_2 può richiedere al max 9 unità
- All'istante t_0 abbiamo il seguente scenario:

| | Richieste massime | Unità possedute |
|-------|-------------------|-----------------|
| P_0 | 10 | 5 |
| P_1 | 4 | 2 |
| P_2 | 9 | 2 |

- All'istante t_0 il sistema è in uno stato sicuro?

Esempio

- Restano altre 3 risorse disponibili e la sequenza $\langle P_1, P_0, P_2 \rangle$ è sicura poiché:
 - ▣ Al processo P_1 possiamo assegnare 2 risorse che poi saranno disponibili e quindi ottenere in seguito 5 risorse disponibili
 - ▣ Al processo P_0 possiamo assegnare 5 risorse che poi saranno disponibili e quindi ottenere in seguito 10 risorse disponibili
 - ▣ Al processo P_2 possiamo assegnare 5 risorse e quindi ottenere al termine tutte le risorse libere

| | Richieste massime | Unità possedute |
|-------|-------------------|-----------------|
| P_0 | 10 | 5 |
| P_1 | 4 | 2 |
| P_2 | 9 | 2 |

Esempio

- Se all'istante t_1 al processo P_2 viene assegnata una sola risorsa disponibile il sistema non è più in uno stato sicuro perché?
 - ▣ Restano altre 2 risorse disponibili da assegnare che possono essere assegnate solo a P_1 ed ottenere al termine 4 risorse disponibili che non soddisfano nessuna delle richieste di P_0 e P_2

| | Richieste massime | Unità possedute |
|-------|-------------------|-----------------|
| P_0 | 10 | 5 |
| P_1 | 4 | 2 |
| P_2 | 9 | 2+1 |

Evitare lo stato non sicuro



- L'idea è quindi di evitare che il sistema finisca in uno stato non sicuro
 - ▣ All'inizio il sistema si trova in uno stato sicuro
 - ▣ Ad ogni richiesta di una risorsa il sistema deve verificare che l'assegnazione lascia il sistema in uno stato sicuro
 - ▣ Lo svantaggio di questo approccio è che un processo potrebbe essere costretto ad attendere una risorsa anche se disponibile

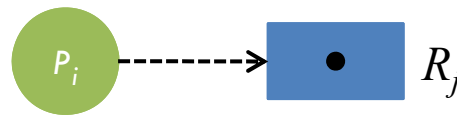
Algoritmi per evitare lo stato non sicuro

1. Algoritmo con grafo di assegnazione delle risorse
 - ▣ Utilizzabile per ogni risorsa esiste una sola istanza
2. Algoritmo del banchiere
 - ▣ Utilizzabile se per ogni risorsa sono possibili più istanze

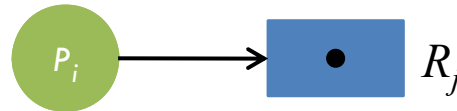
Arco di reclamo

- Nel grafo delle assegnazione delle risorse introduciamo un nuovo tipo di arco chiamato **arco di reclamo**

- Un arco di reclamo $P_i \rightarrow R_j$ indica che il processo P_i può richiedere la risorsa R_j in un qualsiasi momento nel futuro (l'arco è rappresentato come una linea tratteggiata)



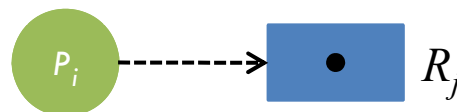
- Quando il processo P_i richiede la risorsa R_j l'arco di reclamo è convertito in un arco di richiesta



- Se la risorsa viene assegnata l'arco di richiesta diventa un arco di assegnazione



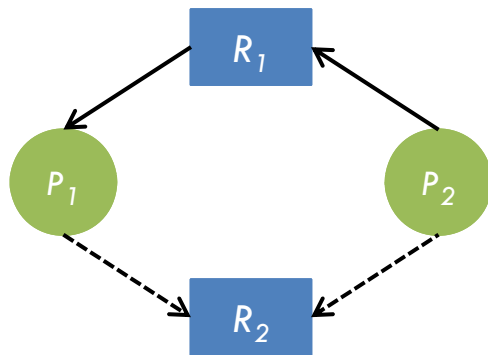
- Al rilascio della risorsa l'arco di assegnazione torna ad essere un arco di reclamo



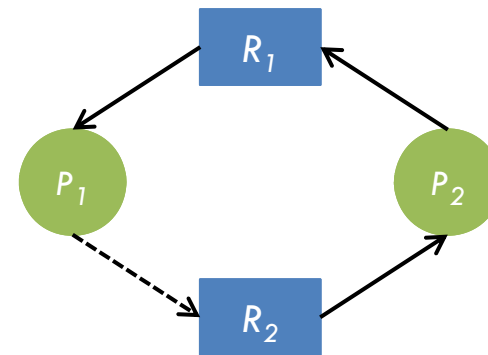
Algoritmo con grafo di assegnazione delle risorse

- Le risorse devono essere reclamate a priori dal sistema
- Conoscendo a priori le risorse che un processo può richiedere è possibile determinare cicli all'interno del grafo prima che si formino effettivamente
- Nel momento in cui un arco di reclamo si converte in un arco di richiesta è possibile verificare se all'interno del grafo si formano cicli e quindi se la risorsa lascia il sistema in un stato sicuro

Esempio



Il processo P_2 richiede la risorsa R_2 che attualmente è libera



La risorsa non può essere assegnata in quanto si creerebbe un ciclo portando il sistema in uno stato non sicuro

Algoritmi per evitare lo stato non sicuro

1. Algoritmo con grafo di assegnazione delle risorse
 - ▣ Utilizzabile per ogni risorsa esiste una sola istanza
2. Algoritmo del banchiere
 - ▣ Utilizzabile se per ogni risorsa sono possibili più istanze

Algoritmo del banchiere

- Supponiamo di avere n processi ed m risorse, definiamo le strutture dati necessarie:
 - ▣ Un vettore **Disponibili** di lunghezza m per ciascun tipo di risorsa
 - $Disponibili[j] = k$ significa che sono disponibili k istanze di risorse R_j
 - ▣ Una matrice **Massimo** di dimensione $m \times n$
 - $Massimo[i,j] = k$ significa che il processo P_i richiede massimo k istanze della risorsa R_j
 - ▣ Una matrice **Assegnate** di dimensione $m \times n$
 - $Assegnate[i,j] = k$ significa che il processo P_i sono assegnate k istanze della risorsa R_j
 - ▣ Una matrice **Necessità** di dimensione $m \times n$
 - $Necessità[i,j] = k$ significa che il processo P_i necessita ancora di k istanze della risorsa R_j
 - $Necessità[i,j] = Massimo[i,j] - Assegnate[i,j]$

Alcune notazioni

- Date due vettori X e Y di lunghezza n diremo che $X \leq Y$ se $X[i] \leq Y[i]$ per ogni $i=1,2,\dots,n$
- Ad esempio se $X=(1,7,3,2)$ e $Y=(0,3,2,1)$ allora $Y \leq X$
- $Y < X$ se $Y \leq X$ e $Y \neq X$

Algoritmo di verifica della sicurezza

- Vogliamo sviluppare un algoritmo che scopre se il sistema è o non è in uno stato sicuro
- Definiamo i seguenti vettori
 - ▣ *Lavoro* di lunghezza m mantiene per ogni risorsa il numero di istanze ancora disponibili
 - ▣ *Fine* di lunghezza n mantiene traccia dello stato di richiesta di ogni processo

Algoritmo di verifica della sicurezza

1. Fase di inizializzazione
 - ▣ $Lavoro = Disponibili$
 - ▣ $Fine[i] = false$
2. Cerca un indice i tale che entrambe le condizioni sono vere
 - ▣ $Fine[i] = false$
 - ▣ $Necessità_i \leq Lavoro$

se tale i non esiste vai al passo 4
1. Imposta $Lavoro = Lavoro + Assegnate_i$
 1. $Fine[i] = vero$
 2. Torna al passo 2
2. Se $Fine[i] == vero$ per ogni i allora il sistema è in uno stato sicuro

Esempio

- Quattro processi P_0, P_1, P_2, P_3 e P_4
- Tre risorse A con 10 istanze, B con 5 istanze, C con 7 istanze
- All'istante T_0 abbiamo la seguente situazione

| | <i>Assegnate</i> | | | <i>Massimo</i> | | | | <i>Necessità</i> | | | <i>Disponibili</i> | | |
|-------|------------------|----------|----------|----------------|----------|----------|-------|------------------|----------|----------|--------------------|----------|----------|
| | <i>A</i> | <i>B</i> | <i>C</i> | <i>A</i> | <i>B</i> | <i>C</i> | | <i>A</i> | <i>B</i> | <i>C</i> | <i>A</i> | <i>B</i> | <i>C</i> |
| P_0 | 0 | 1 | 0 | 7 | 5 | 3 | P_0 | 7 | 4 | 3 | 3 | 3 | 2 |
| P_1 | 2 | 0 | 0 | 3 | 2 | 2 | P_1 | 1 | 2 | 2 | | | |
| P_2 | 3 | 0 | 2 | 9 | 0 | 2 | P_2 | 6 | 0 | 0 | | | |
| P_3 | 2 | 1 | 1 | 2 | 2 | 2 | P_3 | 0 | 1 | 1 | | | |
| P_4 | 0 | 0 | 2 | 4 | 3 | 3 | P_4 | 4 | 3 | 1 | | | |

- Esiste una sequenza sicura che soddisfa i criteri di sicurezza?

$\langle P_1, P_2, P_3, P_4, P_5 \rangle$

Algoritmo di richiesta delle risorse

- Vogliamo sviluppare un algoritmo che verifica se le richieste di un processo possano essere garantite lasciando il sistema in uno stato sicuro
- Definiamo il vettore $Richieste_i$ per indicare il numero di richieste per il processo P_i
 - ▣ Se $Richieste_i[j]=k$ allora il processo P_i richiede k istanza della risorsa R_j
 - ▣ Ad esempio $Richieste_1 = (1,0,2)$ indica che il processo P_1 richiede 1 istanza della risorsa A , 0 della risorsa B e 2 della risorsa C

Algoritmo di richiesta delle risorse

- Se il processo P_i fa una richiesta di risorse allora
 1. Se $Richieste_i \leq Necessità_i$ vai al passo 2 altrimenti il processo richiede troppe risorse
 2. Se $Richieste_i \leq Disponibili_i$ vai al passo 3 altrimenti attendi che le risorse siano disponibili
 3. Simula l'assegnazione delle risorse al processo P_i
 1. $Disponibili = Disponibili - Richieste_i$
 2. $Assegnate_i = Assegnate_i + Richieste_i$
 3. $Necessità_i = Necessità_i - Richieste_i$

Se lo stato di assegnazione delle risorse è sicuro la transazione è completata altrimenti P_i deve attendere $Richieste_i$

Esempio

- Supponiamo che il processo P_1 richieda le seguenti risorse $Richieste_1=(1,0,2)$

| | <i>Assegnate</i> | | | <i>Massimo</i> | | | <i>Necessità</i> | | | <i>Disponibili</i> | | | |
|-------|------------------|----------|----------|----------------|----------|----------|------------------|----------|----------|--------------------|----------|----------|---|
| | <i>A</i> | <i>B</i> | <i>C</i> | <i>A</i> | <i>B</i> | <i>C</i> | <i>A</i> | <i>B</i> | <i>C</i> | <i>A</i> | <i>B</i> | <i>C</i> | |
| P_0 | 0 | 1 | 0 | 7 | 5 | 3 | P_0 | 7 | 4 | 3 | 3 | 3 | 2 |
| P_1 | 2 | 0 | 0 | 3 | 2 | 2 | P_1 | 1 | 2 | 2 | | | |
| P_2 | 3 | 0 | 2 | 9 | 0 | 2 | P_2 | 6 | 0 | 0 | | | |
| P_3 | 2 | 1 | 1 | 2 | 2 | 2 | P_3 | 0 | 1 | 1 | | | |
| P_4 | 0 | 0 | 2 | 4 | 3 | 3 | P_4 | 4 | 3 | 1 | | | |

- Possiamo assegnare le risorse al processo P_1 senza che il sistema entri in uno stato non sicuro?

Rilevamento delle situazioni di stallo

- Se il sistema non può prevenire o evitare situazioni di stallo allora il sistema può ad un certo punto trovarsi in uno stallo
- In questo scenario è necessario:
 - ▣ Un algoritmo per esaminare lo stato del sistema e verificare la presenza di uno stallo
 - ▣ Un algoritmo che ripristini il sistema dalla condizione di stallo
- In ogni caso il lavoro di rilevamento e di ripristino da una situazione di stallo ha costi considerevoli

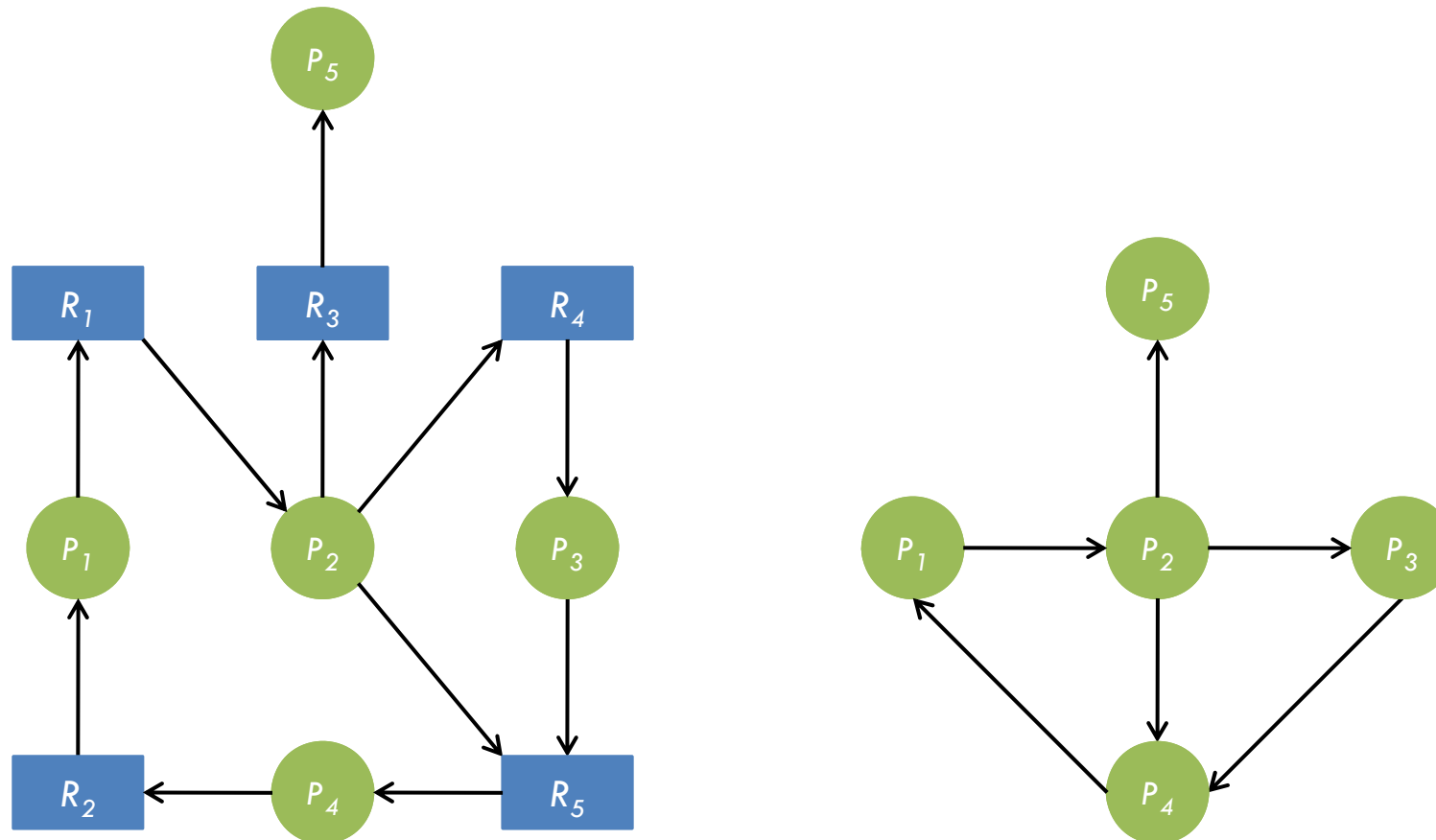
Istanza singola di ciascun tipo di risorsa

- Se tutte le risorse hanno una sola istanza è possibile adoperare il grafo di assegnazione delle risorse e convertirlo in un **grafo di attesa**
- Un grafo di attesa è ottenuto dal grafo di assegnazione delle risorse eliminando le risorse ed aggiungendo archi nel seguente modo
 - ▣ Se il grafo possiede un arco da $P_i \rightarrow R_q$ e da $R_q \rightarrow P_k$ il nuovo grafo avrà un arco da $P_i \rightarrow P_k$



- Nel sistema esisterà uno stallo se il grafo di attesa contiene un ciclo
 - ▣ L'algoritmo deve essere invocato periodicamente
 - ▣ Il numero di operazioni per verificare la presenza di un ciclo all'interno di un grafo con n processi è n^2

Esempio di grafo di attesa



Ripristino da situazioni di stallo

- Nel momento in cui si presenta uno stallo bisogna prendere delle decisioni su come risolverlo
- L'amministratore di sistema potrebbe essere avvisato e quindi lasciare a lui la decisione sul da farsi (ma se avviene il 15 di agosto?)
- Nel caso in cui si vuole procedere automaticamente sono possibili due soluzioni:
 1. Terminare uno o più processi coinvolti nello stallo
 2. Effettuare una prelazione su uno o più processi presenti nello stallo

Più istanze per ciascun tipo di risorsa

- Descriviamo un algoritmo che rileva una situazione di stallo nei sistemi con più istanze per ciascun tipo di risorsa
- Supponiamo di avere m processi ed n risorse, definiamo le strutture dati necessarie:
 - ▣ Un vettore *Disponibili* di lunghezza m per ciascun tipo di risorsa
 - $Disponibili[j] = k$ significa che sono disponibili k istanze di risorse R_j
 - ▣ Una matrice *Assegnate* di dimensione $m \times n$
 - $Assegnate[i,j] = k$ significa che il processo P_i sono assegnate k istanze della risorsa R_j
 - ▣ Una matrice *Richieste* di dimensione $m \times n$
 - $Richieste[i,j] = k$ significa che il processo P_i richiede altre k istanze della risorsa R_j

Algoritmo di rilevamento con più risorse

1. Fase di inizializzazione
 - ▣ $Lavoro = Disponibili$
 - ▣ Se $Assegnate_i \neq 0$ allora $Fine[i] = false$
Altrimenti $Fine[i] = true$
2. Cerca un indice i tale che entrambe le condizioni sono vere
 - ▣ $Fine[i] == false$
 - ▣ $Richieste_i \leq Lavoro$se tale i non esiste vai al passo 4
3. Imposta $Lavoro = Lavoro + Assegnate_i$
 - ▣ $Fine[i] = vero$
 - ▣ Torna al passo 2
4. Se $Fine[i] == false$ per qualche $0 \leq i \leq n$ allora il sistema è in stallo, inoltre, se $Fine[i] == false$ il processo P_i è in stallo

Terminazione di processi

- La terminazione dei processi può avvenire in due modi:
 - ▣ Terminazione di tutti i processi in stallo
 - ▣ Terminazione di un processo alla volta fino all'eliminazione dello stallo
- Se un processo deve essere terminato conviene scegliere un processo di costo minimo rispetto ad alcuni parametri
 - ▣ Priorità dei processi
 - ▣ Tempo trascorso dall'inizio della computazione
 - ▣ Numero di risorse impiegate dal processo
 - ▣ Quantità di risorse ancora in attesa da parte di un processo
 - ▣ Tipi di processi: interattivi o a lotti

Prelazione su risorse

- Prelazionare una risorsa significa sottrarla ad un processo ed assegnarla ad un altro processo che ne fa richiesta
- I problemi da affrontare nella scelta della risorsa da prelazionare sono:
 - ▣ Selezione di una vittima
 - La risorsa da selezionare deve minimizzare i costi di prelazione della stessa
 - ▣ Ristabilimento di un precedente stato sicuro
 - Cosa fare con un processo che non possiede più una risorsa? Normalmente il processo andrebbe riportato in uno stato sicuro. La soluzione più semplice è terminare anche il processo.
 - ▣ Attesa indefinita
 - Se la risorsa è sottratta ad un processo bisogna garantire che non venga scelto sempre lo stesso processo