

## DEADLOCK

**Definizione:** Un insieme di processi bloccati si trova in una situazione di DEADLOCK (stallo) se ogni processo dell'insieme aspetta un evento che solo un altro processo di tale insieme può provocare.

Nella maggior parte dei casi, l'evento che un processo bloccato aspetta è proprio il rilascio di una risorsa da parte di un altro processo bloccato.

Poiché tutti i processi stanno aspettando, nessuno di essi potrà mai causare l'evento che potrebbe far ripartire un altro processo, e così tutti aspettano all'infinito.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

### Esempio di Stallo Hardware:

Supponiamo di avere due processi che vogliono registrare su di un CD un documento digitalizzato da uno scanner:

- 1) Il processo P1 chiede il permesso di utilizzare lo scanner ed è autorizzato;
- 2) Il processo P2 chiede il permesso di utilizzare il masterizzatore ed è autorizzato;
- 3) A questo punto, P1 chiede l'uso del masterizzatore, ovviamente la richiesta viene rifiutata e il processo P1 viene bloccato;
- 4) Successivamente il processo P2 fa una richiesta d'uso dello scanner che ovviamente viene rifiutata, ponendo P2 in blocco.

Dopo questi 4 passi i due processi sono in DEADLOCK (stallo), P1 e P2 sono bloccati poiché ognuno di essi necessita della risorsa allocata all'altro.

### Esempio di Stallo Software:

Nei sistemi che gestiscono basi di dati si possono verificare situazioni di stallo sui processi che gestiscono la lettura e la scrittura.

Supponiamo che due processi P1 e P2 stiano referenziando due record R1 e R2 e quindi hanno bloccato tali risorse, successivamente i due processi richiedono tali record in modo incrociato, ossia il processo P1 richiede R2 e P2 richiede R1, ovviamente come nel caso di prima i due processi sono in STALLO.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I DEADLOCK sono associati al concetto di risorsa disponibile.

Definiamo Risorsa un qualsiasi oggetto Hardware o Software che sia in grado di suscitare interesse in un certo istante di tempo ad un processo. (stampanti, monitor, file, record,etc).

### Risorse Preemptive:

*risorsa che può essere tolta al processo senza provocarne crisi.*

Esempio: la memoria centrale è una risorsa **Preemptive**.

Siano dati due processi P1 e P2 ed entrambi devono stampare un documento

- Il processo P1 ha il possesso della stampante e inizia a stampare, dopo il suo *quanto di tempo* viene messo in blocco.
- P2 richiede la stampante ma questa è stata assegnata a P1 quindi non può stampare.

Situazione di stallo (P2 non può stampare e P1 per stampare ha bisogno della memoria)

- P1 può utilizzare la risorsa memoria in quanto essa è preemptive, essa verrà sottratta al processo P2 e destinata a P1 senza creargli crisi di elaborazione.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

### Risorse Non-Preemptive:

*risorse che **NON** possono essere allocate ad altri processi senza provocare la crisi del processo a cui è stata sottratta.*

Esempio: se un processo P1 ha iniziato a scrivere su di un CD e lo si priva improvvisamente del masterizzatore, il CD sarà incomprensibile in lettura.

**NOTE:** 1. In Generale i deadlock o stalli si hanno con le risorse **NON-preemptive**  
 2. Per le risorse preemptive è possibile risolvere lo stallo.

## Condizione di Stallo(Coffman 1971)

Una situazione di stallo può verificarsi se si verificano contemporaneamente le seguenti 4 condizioni:

- 1) **Mutua esclusione**: ogni risorsa è assegnata ad un solo processo oppure è disponibile.
- 2) **Prendi e Aspetta** (hold and wait): i processi che già hanno richiesto ed ottenuto delle risorse ne possono richiedere altre per essere eseguiti.
- 3) **NON-preemptive**: le risorse che sono state già assegnate a un processo non gli possono essere tolte in modo forzato, ma devono essere rilasciate spontaneamente dal processo che li detiene.
- 4) **Attesa circolare**: deve esistere una catena circolare di processi, ognuno dei quali aspetta il rilascio di una risorsa da parte di un altro processo.

[Go 4 Strategie >](#)

[Go Prevenire >](#)

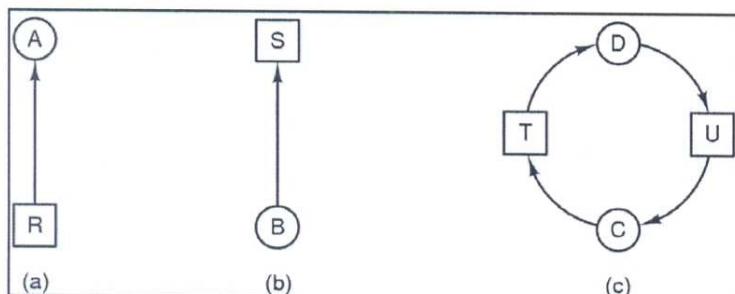
Nota: Queste 4 condizioni devono essere TUTTE presenti affinché ci sia una situazione di stallo, se una di esse non è presente allora nessuna situazione di stallo sarà presente nel sistema.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Modelli per le Situazioni di Stallo(Holt 1972)

Le situazioni di stallo possono essere codificate utilizzando GRAFI ORIETATI con nodi e archi definiti come segue:

- nodi CIRCOLARI rappresentano i processi      
- nodi QUADRATI rappresentano le risorse      
- archi(risorsa, processo): la risorsa è stata assegnata al processo dopo che essa è stata richiesta dallo stesso.      
- archi(processo, risorsa): il processo è bloccato in attesa della risorsa.      



Nota: nel grafo la presenza di un ciclo indica un DEADLOCK che coinvolge i processi e le risorse del ciclo.

C-T-D-U rappresenta uno stallo

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12



In questo esempio i tre processi (A,B,C) richiedono sequenzialmente le risorse R,S,T senza provocare DEADLOCK in quanto sono sequenziali.

NOTA: Il S.O. può mandare in esecuzione un qualsiasi processo non bloccato finché non avrà finito il suo compito.

**La gestione sequenziale non porta a situazioni di stallo** in quanto non è presente nessuna forma di parallelismo con conseguente non competizione sulle risorse.

Nella gestione sequenziale dei processi **i tempi dedicati a effettuare I/O non vengono usati da altri processi per effettuare computo** (uso della CPU), quindi una politica sequenziale non è in generale una buona soluzione.

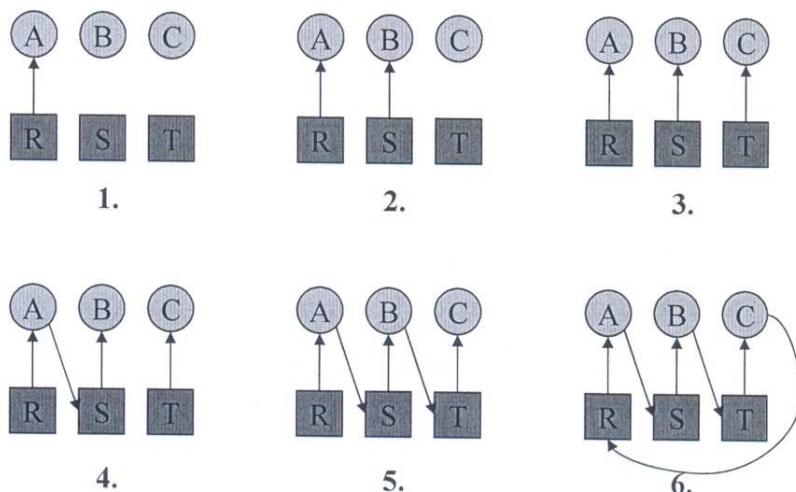
La politica di eseguire processi piccoli che non fanno I/O in una gestione sequenziale dei processi può essere **un'alternativa al round robin**.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

### Esempio di Stallo

1. (A) Richiede R
2. (B) Richiede S
3. (C) Richiede T
4. (A) Richiede S
5. (B) Richiede T
6. (C) Richiede R

**DEADLOCK**



Il S.O. non è obbligato a eseguire i processi come mostrato in questo esempio; se assegnare una risorsa ad un processo può portare a una situazione di stallo allora il S.O. può sospendere il processo e assegnare la risorsa successivamente.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Strategie per le Situazione di Stallo

In generale, per trattare le situazioni di stallo si possono adottare le seguenti strategie:

- 1) Ignorare semplicemente il problema.
- 2) Rilevare e risolvere.
- 3) Cercare dinamicamente di evitare le situazioni di stallo con una accorta politica di allocazione delle risorse.
- 4) Prevenire la situazione di stallo, negando una delle quattro condizioni necessarie [1..4].

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## L'algoritmo dello Struzzo

(Ignorare il problema del Stallo)

L'algoritmo dello Struzzo (non fare assolutamente niente) viene fuori per rispondere ad un quesito semplice ma di complessa soluzione:

### Correttezza o Convenienza, per l'esecuzione dei processi?

**Correttezza:** I Matematici, dietro le loro perversioni mentali, trovano inaccettabile l'esistenza di situazioni di STALLO e quindi sostengono che devono essere necessariamente eliminate.

**Convenienza:** Gli ingegneri di contro, trovano ingiustificato correggere un errore che si verifica raramente (e.i. ogni 5 anni) rispetto ai crash giornalieri di una macchina.

In un elaboratore esistono innumerevoli potenziali situazioni di stallo.

Pensate per esempio alle tabelle gestite da S.O. (tabelle dei processi, tabelle dei file system,...) queste poiché sono di dimensione finita possono creare una condizione di Stallo quando sono piene e un processo tenta di farne richiesta.

In generale l'approccio adottato dai S.O. è quello di **ignorare** il problema, Unix e Windows adottano tale strategia, in quanto tale soluzione è più accettabile di quella in cui ogni utente è costretto ad usare una singola risorsa alla volta.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Rilevare e Risolvere

Piuttosto che prevenire la formazione dello Stallo, questa strategia si preoccupa di garantire la rilevazione dello stallo e di risolvere il problema.

- Identificazione dello stallo quando si è in presenza di un'unica risorsa per tipo.
- Identificazione dello stallo quando si è in presenza di risorse multiple per tipo.

Identificata la situazione di stallo si procede alla sua eliminazione.

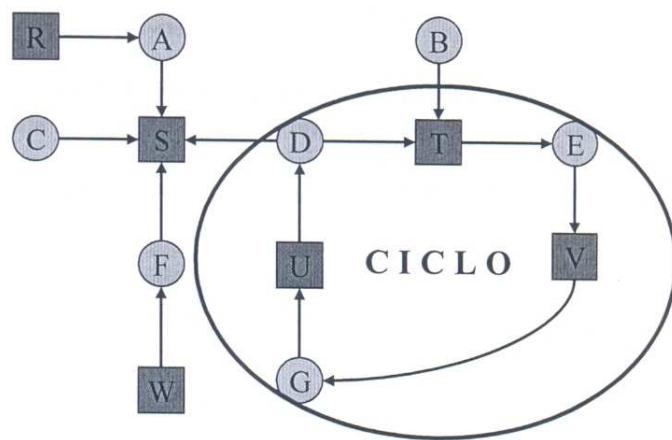
Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Identificazione del Deadlock (Una Risorsa per Tipo)

Consideriamo un sistema in cui esiste un'unica risorsa per ogni tipo di classe (uno scanner, una stampante, etc...). In questo caso il sistema si costruirà un grafo dei processi e delle risorse e valuterà l'esistenza di cicli.

In letterature sono proposti algoritmi per l'individuazione di cicli in un grafo orientato e planare.

1. (A) Occupa R Richiede S
2. (B) Richiede T
3. (C) Richiede S
4. (D) Occupa U Richiede S T
5. (E) Occupa T Richiede V
6. (F) Occupa W Richiede S
7. (G) Occupa V Richiede U



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Algoritmo per la ricerca di cicli in un grafo orientato

**Struttura di dati:** una lista lineare contenente i nodi;

**Azione:** marcare i nodi.

**Ricerca:** Anticipate.

- 1) Per ogni nodo del grafo N, eseguire i seguenti passi con N come radice.
- 2) Inizializzare la lista L come lista vuota e gli archi come non marcati.
- 3) Aggiungere il nodo corrente alla fine di L e controllare se esiste in L, se tale nodo già esiste si è trovato un ciclo e l'algoritmo ha termine.
- 4) Se tutti gli archi uscenti, del nodo corrente, sono marcati allora vai in 6
- 5) Prendere a caso un arco uscente non marcato, marcarlo e considerare il relativo nodo di arrivo come nodo corrente ed andare al passo 3.
- 6) Si è giunti a una foglia(pozzo), si ritorna al nodo precedente, lo si considera nodo corrente e si va al passo 3 se il dono è diverso dal nodo iniziale. Se il nodo è quello iniziale l'algoritmo termina e il grafo non contiene cicli.

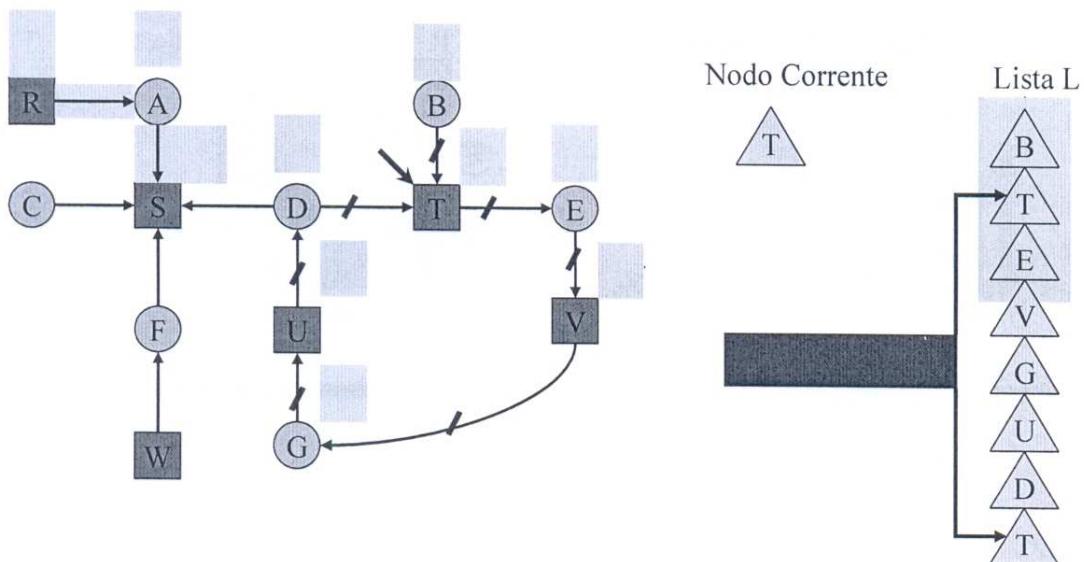
NOTA: l'algoritmo deve essere eseguito su tutti i nodi del grafo.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Algoritmo per la ricerca di cicli in un grafo orientato

### Esempio

Consideriamo l'esempio di prima, si decide di visitare da sinistra verso destra dall'alto verso il basso, facendo partire l'algoritmo da R e successivamente visitando i nodi A,B,C,S,D,T,E,F,...,l'algoritmo ha termine quando si incontra un ciclo.



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Identificazione del Deadlock

(Risorse Multiple per Tipo)

In un sistema di elaborazione definiamo risorsa multipla l'insieme delle risorse afferenti alla stessa classe.

Esempio. Due stampanti, tre lettori di nastri, etc....

Un algoritmo per identificare un deadlock di  $n$  processi in presenza di  $m$  classi di risorse distinte, in cui  $E_i$  rappresenta il numero delle risorse appartenenti alla  $i$ -esima classe, è basato su 4 strutture di dato di tipo matrice.

Strutture di dato necessarie:

Sia **E**(Exist) il vettore di **risorse esistenti** con la cardinalità di  $E$  uguale al numero delle classi ( $m$ ), quindi se la classe 1 identifica le stampanti, allora  $E_1=3$  implica che il sistema ha 3 stampanti.

Sia **A**(Available) il vettore delle **risorse disponibili**, quindi  $A_i$  rappresenta il numero delle risorse disponibili dell' $i$ -esima classe, ossia  $A_1=2$  significa che delle 3 risorse esistenti una è stata assegnata e le altre due sono disponibili.

Sia **C**(Closed) la matrice delle **risorse assegnate**, ogni riga della matrice identifica un vettore relativo ad un processo (iesima riga per iesimo processo) e ogni sua colonna identifica la specifica classe delle risorse assegnate al processo riga.

Sia **R**(Request) la matrice delle **richieste**, ogni riga della matrice è un vettore contenente le richieste di risorse per ogni classe.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Identificazione dello Deadlock

(Risorse Multiple per Tipo)

Esempio:

$E =$  risorse esistenti.

1	•	•	m
---	---	---	---

$A =$  risorse disponibili.

1	•	•	m
---	---	---	---

$C =$  Matrice delle risorse assegnante

Processi ( $n$ )	1	•	•	m
1	•	•	m	
1	•	•	m	
1	•	•	m	

Classi ( $m$ )

$R =$  Matrice delle richieste

Processi ( $n$ )	1	•	•	m
1	•	•	m	
1	•	•	m	
1	•	•	m	

Classi ( $m$ )

$$\sum_{i=1}^n C_{ij} + A_j = E_j$$

$$\frac{\begin{array}{l} \text{Risorse Assegnate} \\ + \\ \text{Risorse Disponibili} \end{array}}{\text{Risorse Esistenti}}$$

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Identificazione dello Deadlock

(Risorse Multiple per Tipo)

163

**Definizione:** Dati due vettori A e B, essi sono in relazione  $A \leq B$  se e solo se per ogni  $1 \leq i \leq m$  si ha  $A_i \leq B_i$

L'algoritmo del deadlock per risorse multiple si basa sul confronto tra vettori.

1. Considera, nella lista dei processi pronti, un Processo non marcato,  $P_i$ , per cui il vettore dell'iesima riga di R(Richieste) è minore o uguale al vettore A(Disponibili)
2. Se si trova un processo di questo tipo MARCA il processo come eseguito, somma l'iesima riga di C(Assegnate) ad A, e torna la passo 1.
3. Se, tra i processi non marcati non esiste un processo  $P_j$  per cui  $R_j \leq A$ , allora l'algoritmo termina.

E = risorse esistenti.

1	•	•	m
---	---	---	---

A = risorse disponibili.

1	•	•	m
---	---	---	---

C = Matrice delle risorse assegnate

1	•	•	m
1	•	•	m
1	•	•	m
1	•	•	m

Classi (m)

R = Matrice delle richieste

1	•	•	m
1	•	•	m
1	•	•	m
1	•	•	m

Classi (m)

*NOTA: Se al termine dell'algoritmo ci sono processi **NON** marcati questi processi sono in Stallo.*

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

164\_a

## Identificazione dello Deadlock

(Risorse Multiple per Tipo)

Esempio: Cinque processi

Tre classi di risorse(A=7, B=2, C=6)

Il processo 1 occupa una risorsa della classe B,

Il processo 2 occupa due risorse della classe A

Il processo 3 occupa tre risorse della classe A e tre della classe C

Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C

Il processo 5 occupa due risorse della classe C

Configurazione iniziale:

$$C = \begin{bmatrix} A & B & C \\ P_1 & 0 & 1 & 0 \\ P_2 & 2 & 0 & 0 \\ P_3 & 3 & 0 & 3 \\ P_4 & 2 & 1 & 1 \\ P_5 & 0 & 0 & 2 \end{bmatrix}$$

Assegnate

$$R = \begin{bmatrix} A & B & C \\ P_1 & 0 & 0 & 0 \\ P_2 & 2 & 0 & 2 \\ P_3 & 0 & 0 & 0 \\ P_4 & 1 & 0 & 0 \\ P_5 & 0 & 0 & 2 \end{bmatrix}$$

Richieste

$$A = \begin{bmatrix} A & B & C \\ 0 & 0 & 0 \end{bmatrix}$$

I processi presenti nella lista dei processi ready sono  $\langle P_1, P_2, P_3, P_4, P_5 \rangle$ :

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Identificazione dello Deadlock

(Risorse Multiple per Tipo)

Esempio: Cinque processi      Tre classi di risorse(A=7, B=2, C=6)

Il processo 1 occupa una risorsa della classe B,

Il processo 2 occupa due risorse della classe A

Il processo 3 occupa tre risorse della classe A e tre della classe C

Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C

Il processo 5 occupa due risorse della classe C

Configurazione iniziale:

	$A$	$B$	$C$					
$P_1$	0	1	0	$P_1$	$A$			
$P_2$	2	0	0	$P_2$	0			
$P_3$	3	0	3	$P_3$	0			
$P_4$	2	1	1	$P_4$	1			
$P_5$	0	0	2	$P_5$	0			
Assegnate			Richieste					
$C = \begin{bmatrix} A & B & C \end{bmatrix}$								
$R = \begin{bmatrix} A & B & C \end{bmatrix}$								

I processi presenti nella lista dei processi ready sono  $\langle P_1, P_2, P_3, P_4, P_5 \rangle$ :

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Identificazione dello Deadlock

(Risorse Multiple per Tipo)

Esempio: Cinque processi      Tre classi di risorse(A=7, B=2, C=6)

Il processo 1 occupa una risorsa della classe B,

Il processo 2 occupa due risorse della classe A

Il processo 3 occupa tre risorse della classe A e tre della classe C

Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C

Il processo 5 occupa due risorse della classe C

Configurazione iniziale:

	$A$	$B$	$C$					
$P_1$	0	1	0	$P_1$	$A$			
$P_2$	2	0	0	$P_2$	0			
$P_3$	3	0	3	$P_3$	0			
$P_4$	2	1	1	$P_4$	1			
$P_5$	0	0	2	$P_5$	0			
Assegnate			Richieste					
$C = \begin{bmatrix} A & B & C \end{bmatrix}$								
$R = \begin{bmatrix} A & B & C \end{bmatrix}$								

I processi presenti nella lista dei processi ready sono  $\langle P_1, P_2, P_3, P_4, P_5 \rangle$ :

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## **Identificazione dello Deadlock** **(Risorse Multiple per Tipo)**

Esempio: Cinque processi Tre classi di risorse(A=7, B=2, C=6)

- Il processo 1 occupa una risorsa della classe B,
- Il processo 2 occupa due risorse della classe A
- Il processo 3 occupa tre risorse della classe A e tre della classe C
- Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C
- Il processo 5 occupa due risorse della classe C

## Configurazione iniziale:

$$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad C = \begin{bmatrix} A & B & C \\ 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 3 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

Assegnate

$$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad R = \begin{bmatrix} A & B & C \\ 0 & 0 & 0 \\ 2 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Richieste

$$A = \begin{bmatrix} A & B & C \\ 3 & 1 & 3 \end{bmatrix}$$

I processi presenti nella lista dei processi ready sono <P1,P2,P3,P4,P5>:

## **Identificazione dello Deadlock**

**(Risorse Multiple per Tipo)**

Esempio: Cinque processi Tre classi di risorse(A=7, B=2, C=6)

- Il processo 1 occupa una risorsa della classe B,
- Il processo 2 occupa due risorse della classe A
- Il processo 3 occupa tre risorse della classe A e tre della classe C
- Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C
- Il processo 5 occupa due risorse della classe C

## Configurazione iniziale:

$$P_1 \quad C = \begin{bmatrix} A & B & C \\ 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 3 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix} \quad \text{Assegnate}$$

$$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5$$

$$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5$$

$$R = \begin{bmatrix} A & B & C \\ 0 & 0 & 0 \\ 2 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \text{Richieste}$$

$$A = \begin{bmatrix} A & B & C \\ 3 & 1 & 3 \end{bmatrix}$$


I processi presenti nella lista dei processi ready sono <P1,P2,P3,P4,P5>;

## **Identificazione dello Deadlock**

(Risorse Multiple per Tipo)

Esempio: Cinque processi Tre classi di risorse(A=7, B=2, C=6)

Il processo 1 occupa una risorsa della classe B,

Il processo 2 occupa due risorse della classe A

Il processo 3 occupa tre risorse della classe A e tre della classe C.

Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C.

Il processo 5 occupa due risorse della classe C

## Configurazione iniziale:

$$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad C = \begin{bmatrix} A & B & C \\ 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 3 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

Assegnate

$$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad R = \begin{bmatrix} A & B & C \\ 0 & 0 & 0 \\ 2 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \text{Richieste}$$

$$A = \begin{bmatrix} A & B & C \\ 5 & 2 & 4 \end{bmatrix}$$

I processi presenti nella lista dei processi ready sono <P1,P2,P3,P4,P5>;

## **Identificazione dello Deadlock (Risorse Multiple per Tipo)**

Esempio: Cinque processi Tre classi di risorse(A=7 B=2 C=6)

Il processo 1 occupa una risorsa della classe B

Il processo 1 occupa una risorsa della classe B.

Il processo 3 occupa tre risorse della classe A e tre della classe C

Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C.

Il processo 5 occupa due risorse della classe C.

Configurazione iniziale:

$$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad C = \begin{bmatrix} A & B & C \\ 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 3 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

Asseggnate

$$P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad R = \begin{bmatrix} A & B & C \\ 0 & 0 & 0 \\ 2 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \text{Richieste}$$

$$A = \begin{bmatrix} A & B & C \\ 5 & 2 & 4 \end{bmatrix}$$

I processi presenti nella lista dei processi ready sono <P1,P2,P3,P4,P5>;

## Identificazione dello Deadlock (Risorse Multiple per Tipo)

Esempio: Cinque processi      Tre classi di risorse(A=7, B=2, C=6)

- Il processo 1 occupa una risorsa della classe B,
- Il processo 2 occupa due risorse della classe A
- Il processo 3 occupa tre risorse della classe A e tre della classe C
- Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C
- Il processo 5 occupa due risorse della classe C

Configurazione iniziale:

$$\begin{array}{l}
 P_1 \quad \left[ \begin{array}{ccc} A & B & C \\ 0 & 1 & 0 \end{array} \right] \\
 P_2 \quad \left[ \begin{array}{ccc} A & B & C \\ 2 & 0 & 0 \end{array} \right] \\
 P_3 \quad \left[ \begin{array}{ccc} A & B & C \\ 3 & 0 & 3 \end{array} \right] \\
 P_4 \quad \left[ \begin{array}{ccc} A & B & C \\ 2 & 1 & 1 \end{array} \right] \\
 P_5 \quad \left[ \begin{array}{ccc} A & B & C \\ 0 & 0 & 2 \end{array} \right]
 \end{array}
 \quad C = \left[ \begin{array}{cc|c} P_1 & P_2 & R \\ \hline
 0 & 0 & 0 \\
 2 & 0 & 2 \\
 3 & 0 & 0 \\
 2 & 1 & 0 \\
 0 & 0 & 2
 \end{array} \right]
 \quad \begin{matrix} \text{Assegnate} \\ \text{Richieste} \end{matrix}$$

$$A = \left[ \begin{array}{ccc} A & B & C \\ 5 & 2 & 6 \end{array} \right]$$

I processi presenti nella lista dei processi ready sono <P1,P2,P3,P4,P5>:

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Identificazione dello Deadlock (Risorse Multiple per Tipo)

Esempio: Cinque processi      Tre classi di risorse(A=7, B=2, C=6)

- Il processo 1 occupa una risorsa della classe B,
- Il processo 2 occupa due risorse della classe A
- Il processo 3 occupa tre risorse della classe A e tre della classe C
- Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C
- Il processo 5 occupa due risorse della classe C

Configurazione iniziale:

$$\begin{array}{l}
 P_1 \quad \left[ \begin{array}{ccc} A & B & C \\ 0 & 1 & 0 \end{array} \right] \\
 P_2 \quad \left[ \begin{array}{ccc} A & B & C \\ 2 & 0 & 0 \end{array} \right] \\
 P_3 \quad \left[ \begin{array}{ccc} A & B & C \\ 3 & 0 & 3 \end{array} \right] \\
 P_4 \quad \left[ \begin{array}{ccc} A & B & C \\ 2 & 1 & 1 \end{array} \right] \\
 P_5 \quad \left[ \begin{array}{ccc} A & B & C \\ 0 & 0 & 2 \end{array} \right]
 \end{array}
 \quad C = \left[ \begin{array}{cc|c} P_1 & P_2 & R \\ \hline
 0 & 0 & 0 \\
 2 & 0 & 2 \\
 3 & 0 & 0 \\
 2 & 1 & 0 \\
 0 & 0 & 2
 \end{array} \right]
 \quad \begin{matrix} \text{Assegnate} \\ \text{Richieste} \end{matrix}$$

$$A = \left[ \begin{array}{ccc} A & B & C \\ 5 & 2 & 4 \end{array} \right]$$

I processi presenti nella lista dei processi ready sono <P1,P2,P3,P4,P5>:

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Identificazione dello Deadlock

(Risorse Multiple per Tipo)

Esempio: Cinque processi      Tre classi di risorse(A=7, B=2, C=6)

- Il processo 1 occupa una risorsa della classe B,
- Il processo 2 occupa due risorse della classe A
- Il processo 3 occupa tre risorse della classe A e tre della classe C
- Il processo 4 occupa due risorse della classe A, una risorsa di B e una di C
- Il processo 5 occupa due risorse della classe C

Configurazione iniziale:

	$A$	$B$	$C$		$A$	$B$	$C$
$P_1$	0	1	0	$P_1$	0	0	0
$P_2$	2	0	0	$P_2$	2	0	2
$P_3$	3	0	3	$P_3$	0	0	0
$P_4$	2	1	1	$P_4$	1	0	0
$P_5$	0	0	2	$P_5$	0	0	2
Assegnate				Richieste			

**Nota:** tutte le risorse sono state assegnate, ossia nessuna risorsa è disponibile.

I processi presenti nella lista dei processi ready sono  $\langle P_1, P_2, P_3, P_4, P_5 \rangle$ :

Sostituendo le richieste del Processo P3 con [0 0 1] i processi saranno in DEADLOCK

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Quando attivare l'identificazione del Deadlock

- 1) Una possibilità è quella di attivare l'identificazione ogni qual volta si fa una richiesta di risorsa....Questo impiegherebbe un notevole uso della CPU per il controllo(non adeguato).
- 2) Un'altra strategia potrebbe essere quella di attivarlo ogni k minuti;
- 3) Quando l'utilizzo della CPU scende al di sotto di una determinata soglia, in relazione ai processi presenti nel sistema.

## Come Risolvere un Deadlock

- Risoluzione del DEADLOCK mediante Preerilascio

- Risoluzione del DEADLOCK mediante ritorno allo stato precedente

- Risoluzione del DEADLOCK mediante eliminazione del processo

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

### Risoluzione del DEADLOCK mediante Prerilascio

Tale soluzione si adotta in casi estremi e solo quando è possibile rilasciare la risorsa.

**Esempio:** nei sistemi Batch un processo che sta utilizzando una stampante può essere sospeso dalla stampa senza mandare in crisi il processo e più in generale il sistema; la risorsa stampante sarà assegnata ad un altro processo, e successivamente il processo riprenderà la propria stampa.

**Effetto collaterale -** La stampa dei documenti dei due processi non sarà corretta: le pagine dei documenti saranno mescolate.

Un effetto analogo lo si ha con la risorsa memoria centrale, anch'essa rilasciabile dal processo senza causare troppi problemi.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Risoluzione del DEADLOCK mediante ritorno allo stato precedente

177

1. Individuato il Deadlock è facile scoprire quali sono le risorse coinvolte.  
(Algoritmo del grafo o delle matrici svolge tale compito).
1. Individuate le risorse coinvolte nel Deadlock e quindi i relativi processi, il sistema riporta tale processo allo stato precedente ossia prima della richiesta della risorsa.
2. Il compito svolto dal processo sino a quel momento viene abbandonato e si riparte dalla configurazione priva di deadlock.
3. Se il processo ripartito prova ad acquisire di nuovo la nuova risorsa, dovrà attendere finché sarà disponibile in quanto potrebbe essere assegnata ad altro processo.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Risoluzione del DEADLOCK mediante eliminazione del processo

178

Un modo poco ortodosso per risolvere i problemi del Deadlock è l'eliminazione successiva di processi sino ad ottenere, attraverso l'interruzione del ciclo, l'eliminazione del Deadlock.

In alternativa si potrebbe scegliere attentamente come vittima un processo NON appartenente al ciclo, magari con allocate molte risorse. Tale strategia attacca il problema della risoluzione del Deadlock in modo indiretto: anche se il processo non è direttamente coinvolto nel Deadlock, il rilascio di molte risorse, di pertinenza di tale processo, potrebbe agevolare qualche processo posto in blocco.

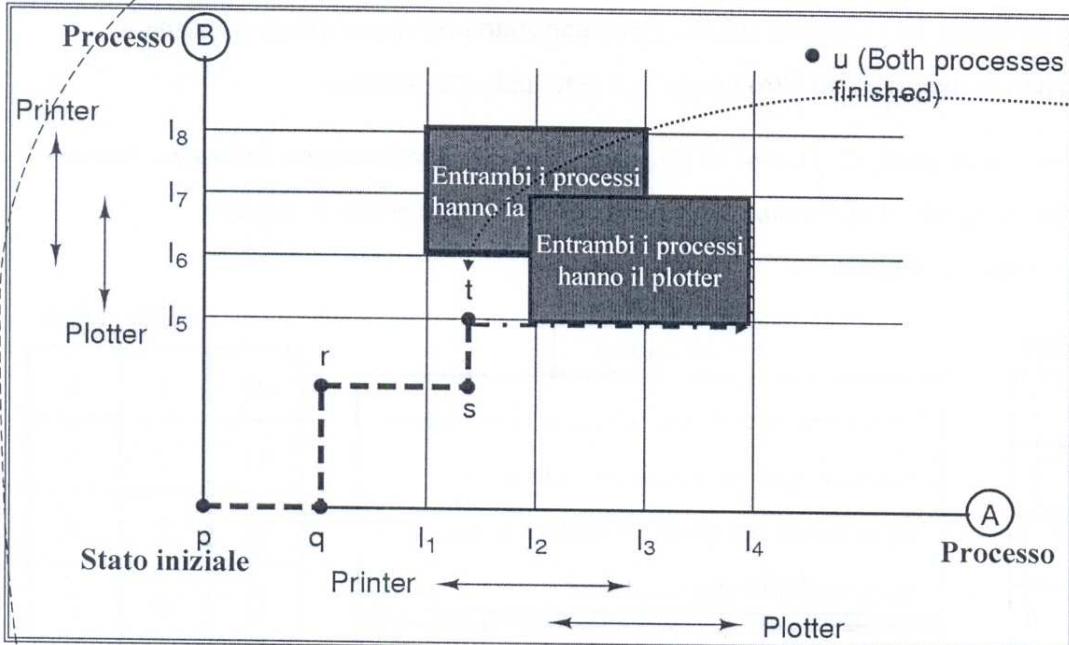
L'eliminazione dei processi deve essere fatta con molta attenzione, in genere è preferibile eliminare quei processi che non causano problemi se rieseguiti dall'inizio.

**Esempio OK:** la compilazione di un sorgente è certamente un processo eliminabile, in quanto la sua riesecuzione non crea nessun problema, il sorgente resta integro e l'eseguibile non subisce variazioni.

**Esempio NOK:** la soppressione di un processo che sta eseguendo un aggiornamento di un database è certamente la meno consigliabile in quanto si rischia di aggiornare due volte lo stesso dato nel database

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Esiste un algoritmo che evita un Deadlock facendo ogni volta la scelta opportuna?



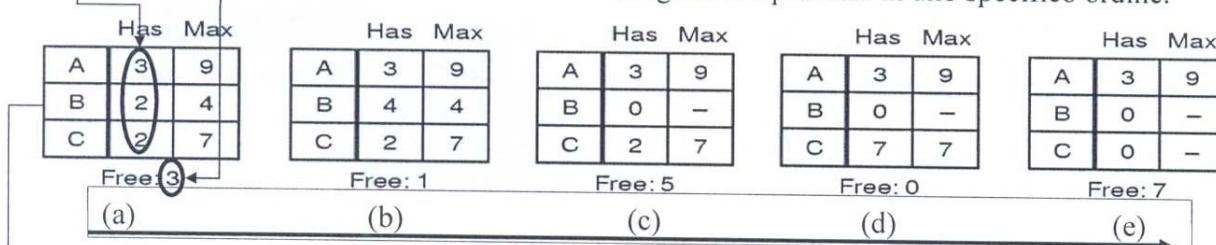
La risposta è affermativa ma solo se sono disponibili in anticipo specifiche informazioni.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

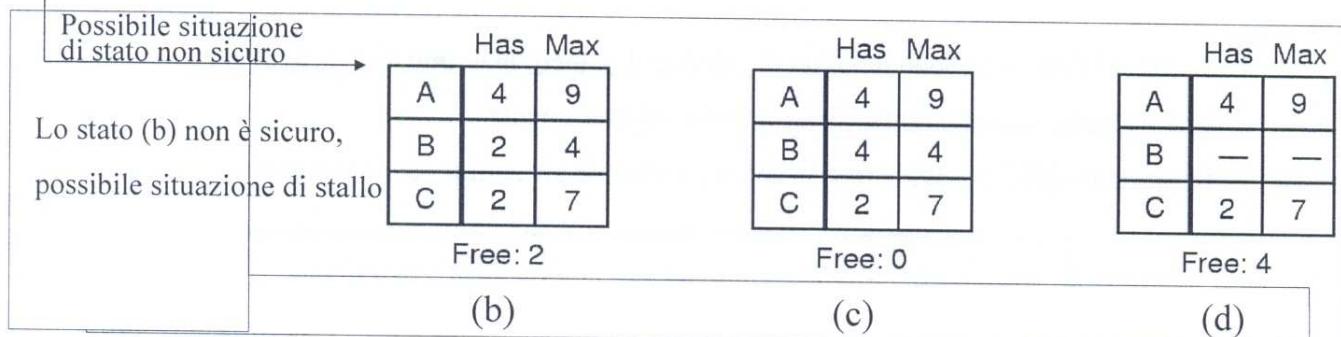
## Come Evitare Deadlock

### Stati Sicuri

**Definizione:** Uno stato è detto sicuro se NON è in stallo e se esiste una strategia per soddisfare tutte le richieste pendenti eseguendo i processi in uno specifico ordine.



Lo stato in (a) è sicuro quindi non porterà a situazioni di stallo.



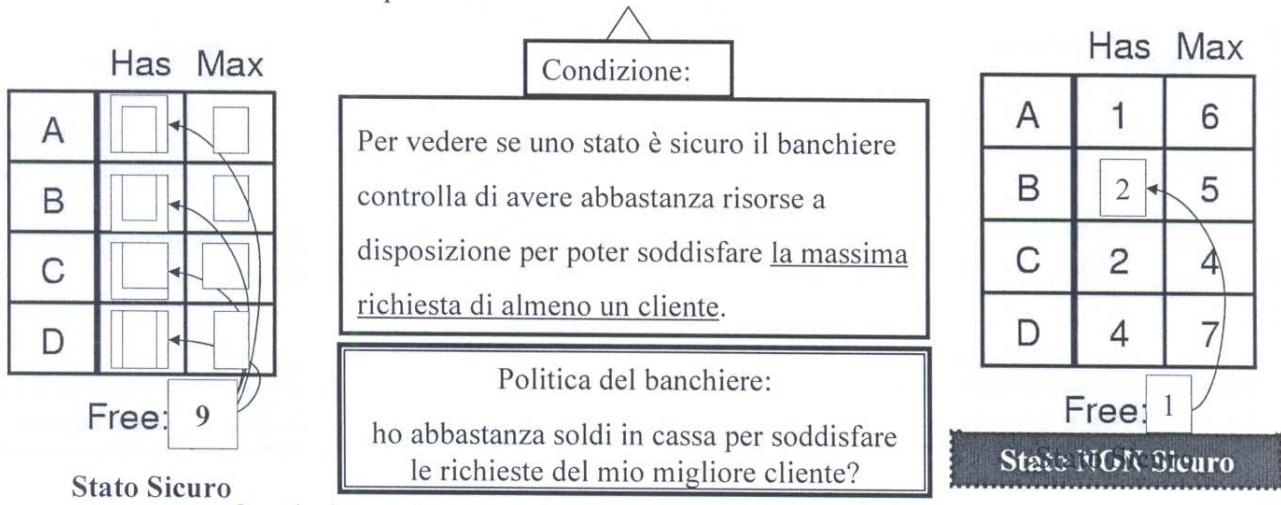
**NOTA:** Uno stato non sicuro non conduce necessariamente ad una situazione di stallo.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Algoritmo del Banchiere per una singola risorsa

Strategia dell'algoritmo del banchiere è di controllare lo stato successivo dopo avere assegnato una risorsa a un processo. Se lo stato in cui si andrà è uno stato non sicuro allora si ritornerà indietro assegnando la risorsa ad un altro processo e reiterando l'algoritmo.

I definitiva prima di assegnare una risorsa ad un processo si fa una valutazione dello stato futuro con quella risorsa assegnata al processo, se lo stato a cui si giunge è sicuro si assegna definitivamente la risorsa al processo.



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Come Evitare Deadlock

### Algoritmo del Banchiere per risorse multiple

$$\begin{array}{l}
 P_1 \quad C = \begin{bmatrix} A & B & C \\ 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 3 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix} \qquad P_1 \quad R = \begin{bmatrix} A & B & C \\ 0 & 0 & 0 \\ 2 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \\
 \text{Risorse assegnate} \qquad \qquad \qquad \text{Risorse richieste}
 \end{array}$$

$A = \begin{bmatrix} A & B & C \\ 7 & 2 & 6 \end{bmatrix}$   
 Risorse disponibili

Note:

1. I processi raramente sono in grado di stabilire il numero massimo di risorse di cui hanno bisogno.
2. Il numero dei processi non è fisso ma varia dinamicamente.
3. Dinamicità delle risorse, le risorse che sono ritenute presenti possono guastarsi nel tempo.

**L'algoritmo del Banchiere per evitare i DEADLOCK è  
raramente usato negli attuali S.O.**

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Evitare un deadlock è praticamente impossibile poiché per far ciò bisogna avere informazioni sulle richieste future di ciascun processo.

Una soluzione può essere data dalla negoziazione delle 4 condizioni necessarie affinché si verifichi un deadlock (coffmann 1971):

- 1) NEGOZIARE la mutua esclusione.
- 2)NEGOZIARE la condizione di hold-and-wait.
- 3)NEGOZIARE la condizione di mancanza di preemptive
- 4)NEGOZIARE la condizione di attesa circolare.

[Go to coffman](#)

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Negoziare la mutua esclusione

Se nessuna risorsa fosse MAI assegnata in maniera esclusiva ad un unico processo, non avremo MAI situazioni di stallo.

**NO non-preemptive!!!!**

Immaginiamo la risorsa stampante: una stampante porterebbe a delle stampe mescolate sia in termini di pagine, sia di paragrafi sia di caratteri all'interno della pagina.

**Soluzione:** definire un processo(spool di stampa), in cui l'unica risorsa richiesta è la stampante, è certamente una soluzione accettabile. I processi che necessitano di stampa non richiedono direttamente la stampante ma fanno riferimento a un demone (spool di stampa) che da una parte riceve i file da stampare e dall'altra li accoda pazientemente sulla stampante.

Nota: con lo spool di stampa non si possono formare deadlock in quanto lo spool richiede soltanto la stampante come risorsa, e nessun processo può richiedere la stampante. Inoltre quando un processo richiede lo spooler e questo è lockato da un altro processo esso sarà posto in wait sino a quando lo spooler sarà liberato, questo avverrà sicuramente in quanto tra i due processi avverrà uno scambio di dati senza richiesta di ulteriori risorse.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Negoziare la mutua esclusione<sub>2</sub>

A questo punto pensiamo che il problema del deadlock, almeno per le stampe, è risolto.....  
MA!!

Anche questa soluzione può portare ad un deadlock:

ESEMPIO: La competizione per lo spazio di spool su disco può portare ad una situazione di stallo.

**Esempio:** due processi che riempiono per metà lo spazio disco dello spool di stampa senza che nessuno abbia finito il proprio lavoro.

Gli spool di stampa sono progettati in modo da stampare solo quando l'intero file è stato scritto sul disco, la strategia di stampare porzioni del file man mano che il file è scritto sul disco non è vincente in quanto dopo la prima porzione di stampa il processo di spool potrebbe attendere molto tempo prima di riprendere la stampa con conseguente inattività della stessa.

### Idea Principale

Evitare di assegnare una risorsa quando non è strettamente necessario ed assicurarsi che la risorsa sia richiesta dal minor numero possibile di processi.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Negoziare la condizione di hold-and-wait

Fare in modo che un processo che detiene alcune risorse NON ne richieda delle altre eviterebbe lo stallo

Soluzione:

Il processo richiede, nella sua fase iniziale e prima della sua esecuzione, tutte le risorse di cui ha necessità.

Tale soluzione garantisce la non richiesta di ulteriori risorse durante l'esecuzione del processo.

Domande:

- 1) è possibile sapere a priori quali sono le risorse di cui necessita il processo?
- 2) l'assegnazione a priori delle risorse ottimizza il loro uso?

Risposta:

in entrambi i casi è No.

Soluzione alternativa:

Quando un processo richiede una o più risorse rilascia tutte le sue risorse richiedendole tutte insieme successivamente.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Negoziare la condizione di mancanza di preemptive

Togliere una risorsa pregiata in condizione di assenza di prerilascio è in molti casi impossibile:

Supponiamo di avere un plotter e una stampante, il processo P1 tenta di stampare sul plotter ma esso non è disponibile in quanto spento, l'alternativa è di stampare su di una stampante che è occupata da un altro processo, se la stampante è con prerilascio si rischia di provocare caos durante la stampa.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Negoziare la condizione di attesa circolare

Come evitare l'attesa circolare:

Possibili Soluzioni:

1. Imporre la regola che ad ogni processo venga assegnata una risorsa alla volta. Quando un processo necessita di una risorsa oltre quella da lui posseduta esso è obbligato a rilasciare la precedente, prima di acquisire la seconda.

Inaccettabile per processi che leggono da file e stampano.

2. Fornire una numerazione globale a tutte le risorse, i processi possono chiedere tutte le risorse che desiderano ma devono rispettare l'ordinamento; ossia, il processo che ha acquisito la risorsa  $i^{esima}$  può acquisire la risorsa  $j^{esima}$  se  $j > i$ .

Esempio: dati due processi A, B con assegnazioni di risorse h e k rispettivamente, allora se  $h < k$ , B non potrà richiedere la risorsa h e contestualmente se  $k < h$  A non potrà richiedere la risorsa k.

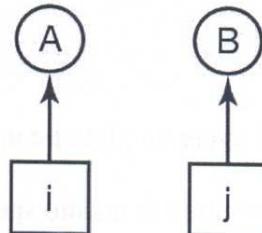
Nota: un processo che rilascia tutte le risorse ricomincia da capo.

Trovare un ordinamento delle risorse che vada bene per tutti i processi non è semplice. Pensare a risorse software (tabelle di processo, spazio sul disco, record di database) il loro numero insieme con il numero dei processi può essere talmente elevato da rendere impossibile l'individuazione di un corretto ordinamento.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Negoziare la condizione di attesa circolare

1. Imagesetter
2. Scanner
3. Plotter
4. Tape drive
5. CD Rom drive



**NOTA:** se un processo richiede una risorsa con indice minore rilascia le risorse possedute con indice maggiore.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

## Approcci per la prevenzione dei DEADLOCK

CONDIZIONE	APPROCCIO
Mutua Esclusione	— - - → Spool di tutto
Hold and Wait	— - - → Richiedere inizialmente tutte le risorse
Prerilascio	— - - → Portare via alcune risorse
Attesa Circolare	— - - → Ordinare numericamente tutte le risorse

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12