Sistemi Operativi

C.d.L. in Informatica (laurea triennale)
Anno Accademico 2011-2012

Dipartimento di Matematica e Informatica – Catania

Deadlock

Prof. Mario Di Raimondo

Risorse di un sistema

- Un processo fa uso di **varie risorse** e di diversi **tipi**:
  - risorse hardware;
  - risorse software;
- la maggior parte di queste devono essere utilizzate in **modalità esclusiva**;
- risorsa **preemptable/non-preemptable**;
- **fasi** di impiego di una risorsa:
  (1) richiesta;
  (2) utilizzo;
  (3) rilascio;
- **blocco** in caso di mancata disponibilità:
- impiego di **semafori mutex** per risorse software.

Il Deadlock

- Vediamo alcuni semplici **esempi**:
  - due processi (A e B) che vogliono utilizzare due risorse (una stampante e uno scanner);
  - tre processi che necessitano di due masterizzatori DVD ciascuno; presenza di tre masterizzatori in totale sul sistema;
- **definizione**:

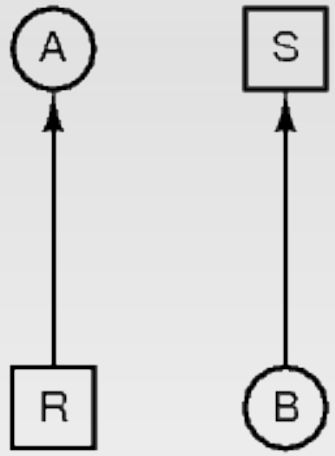
  ”*Un insieme di processi è in stato di **deadlock** se ciascun processo dell'insieme è in attesa di un evento che solo un altro processo dell'insieme può provocare.*”
- noi tratteremo dei **deadlock delle risorse**:
  - tecniche mutuabili anche in altri ambiti.

Condizioni

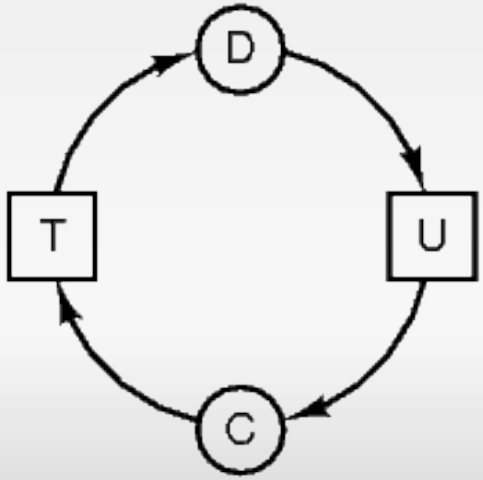
- Esistono 4 condizioni strettamente **necessarie** affinché si verifichi uno stallo:
  - **mutua esclusione**;
  - **possesso e attesa**;
  - **impossibilità di prelazione**;
  - **attesa circolare**;
- lo stallo si può verificare solo se tutte le suddette condizioni sono verificate.

Modellazione

- Possiamo utilizzare dei **grafi orientati** per ragionare sull'allocazione delle risorse ai processi:
  - **grafo delle risorse**:
    - **nodi**: processi e risorse;
    - **archi**:
      - processo → risorsa: richiesta;
      - risorsa → processo: assegnazione;



- un **deadlock** si identifica con un ciclo nel grafo:

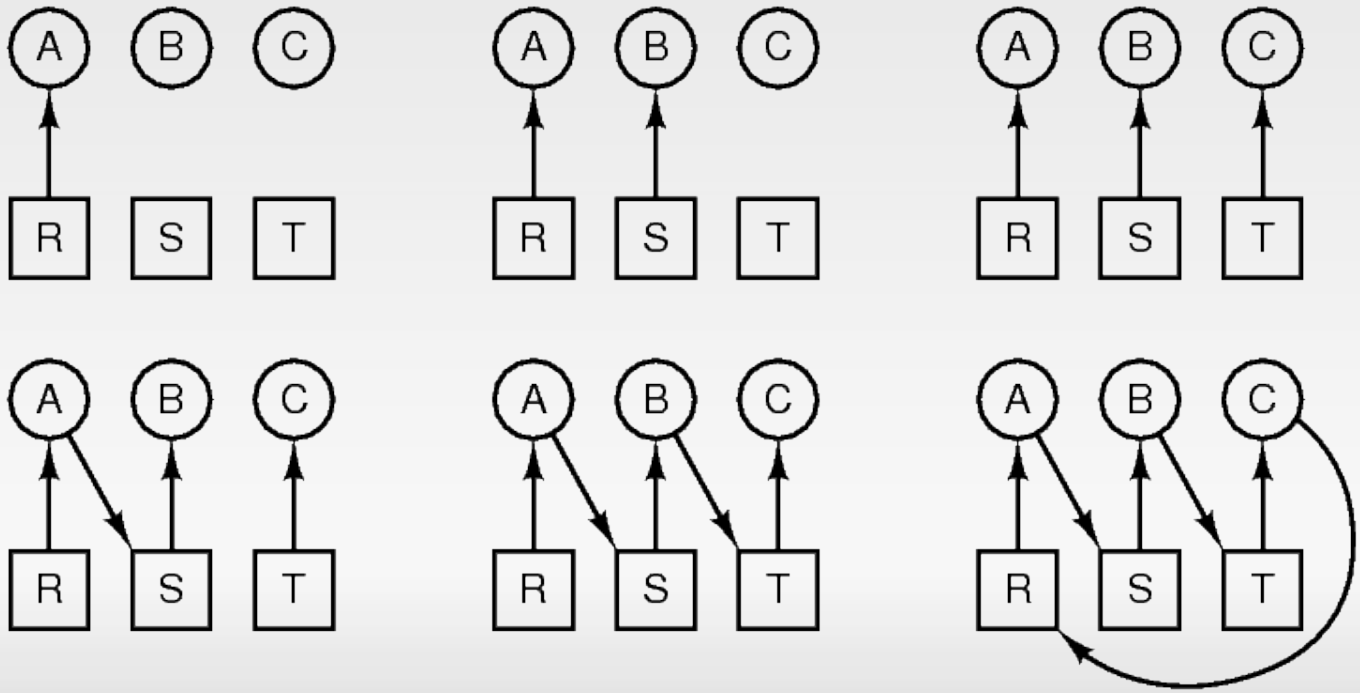


Esempio con grafo

- gestione delle risorse da parte dei processi:
  - A: richiede R; richiede S; rilascia R; rilascia S;
  - B: richiede S; richiede T; rilascia S; rilascia T;
  - C: richiede T; richiede R; rilascia T; rilascia R;
- uno scheduling possibile:

(1) A richiede R;
(2) B richiede S;
(3) C richiede T;
(4) A richiede S;
(5) B richiede T;
(6) C richiede R;

STALLO!



Esempio con grafo

- gestione delle risorse da parte dei processi:
  - A: richiede R; richiede S; rilascia R; rilascia S;
  - B: richiede S; richiede T; rilascia S; rilascia T;
  - C: richiede T; richiede R; rilascia T; rilascia R;
- uno scheduling alternativo:

(1) A richiede R;

(2) C richiede T;

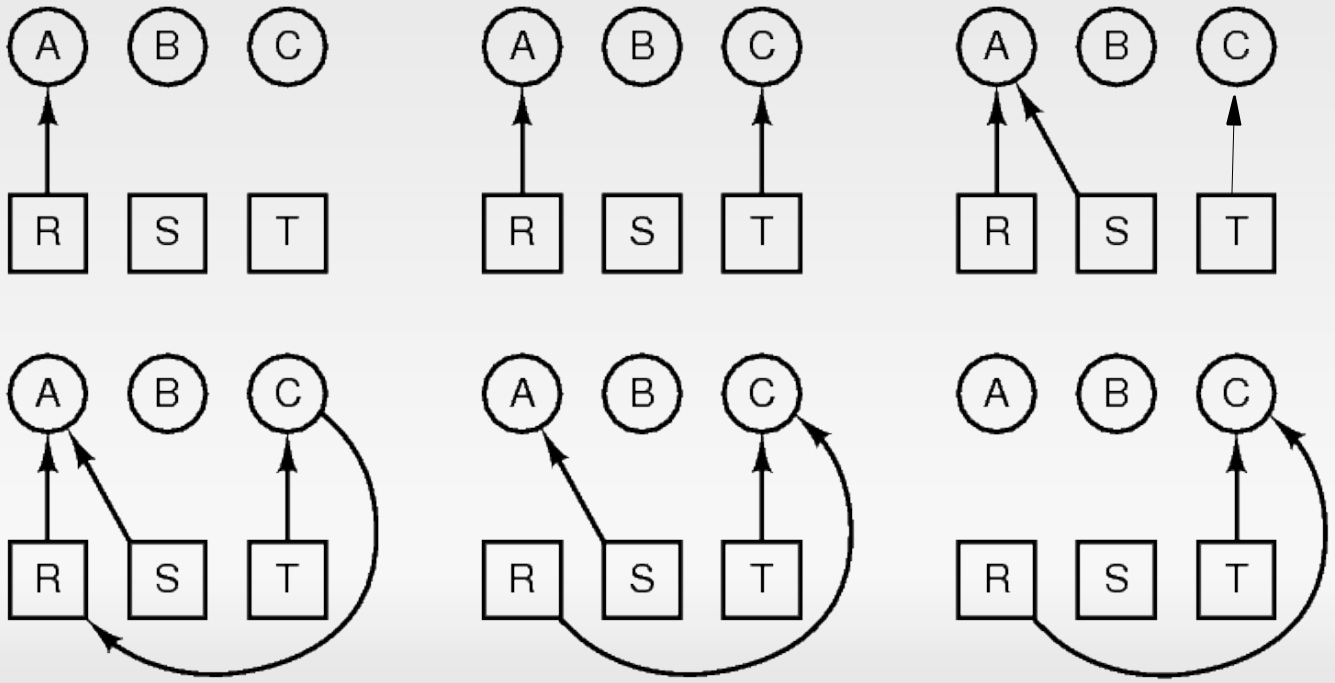
(3) A richiede S;

(4) C richiede R;

(5) A rilascia R;

(6) A rilascia S;

(nessuno stallo)

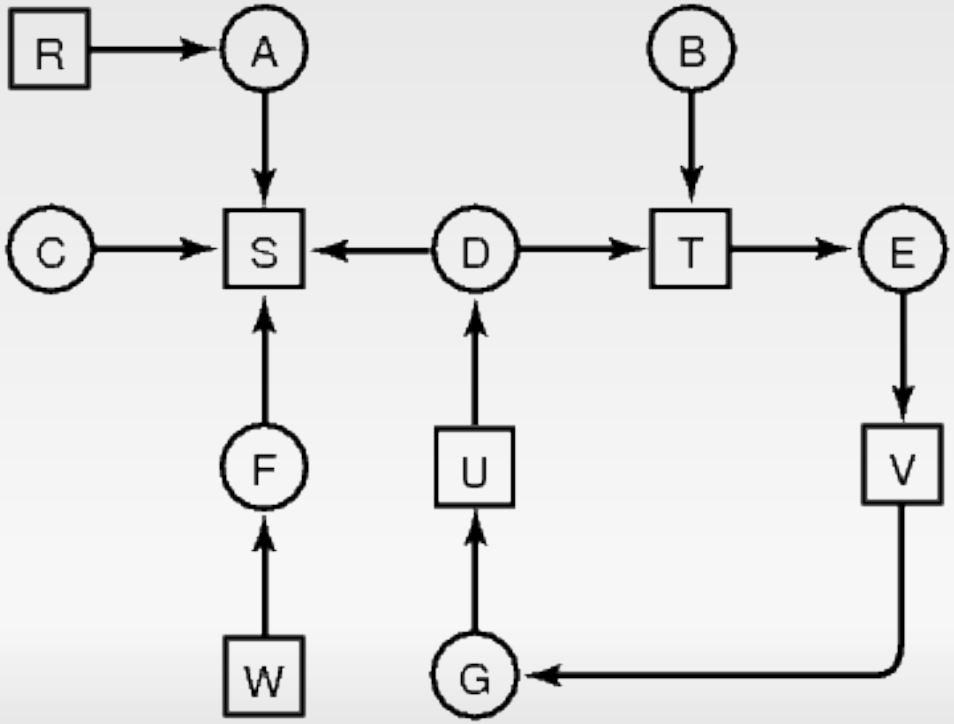


Strategie contro i deadlock

- Le attuali strategie note per gestire in qualche modo gli stalli sono:
  - **ignorare** del tutto l'eventualità che si verifichino;
  - cercare di **rilevare** la presenza di deadlock e **agire** di conseguenza per poterne uscire;
  - **evitare** di compiere azioni che possano portare a deadlock;
  - **prevenire** del tutto il loro verificarsi.

Rilevare gli stalli

- **Caso semplice**: una sola risorsa per ciascun tipo;
- si considera il **grafo delle risorse** e si controlla la presenza di eventuali **cicli**.



Rilevare gli stalli

- **Caso più generale**: *n* processi, *m* classi di risorse;
  - vettore delle **risorse esistenti**: *E*;
  - vettore delle **risorse disponibili**: *A*;
  - matrice dell'**allocazione corrente**: *C*;
  - matrice delle **richieste**: *R*;
- **invariante**: $\sum_{i=1..n} C_{ij} + A_j = E_j$ per ogni possibile risorsa *j*

Resources in existence
$(E_1, E_2, E_3, \ldots, E_m)$

Resources available
$(A_1, A_2, A_3, \ldots, A_m)$

Current allocation matrix

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm} \end{bmatrix}$$

Request matrix

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1m} \\ R_{21} & R_{22} & R_{23} & \cdots & R_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & R_{n3} & \cdots & R_{nm} \end{bmatrix}$$

Rilevare gli stalli

- **algoritmo**:
  - cercare un processo P_i tale che la i-esima riga di R è ≤ A;
    - trovato: aggiungi i-esima riga di C ad A e marca il processo P_i; torna al punto precedente;
    - non trovato: esci;
  - assumiamo caso peggiore: il processo mantiene le risorse fino alla sua terminazione;
  - se rimane qualche processo non marcato, è presente un deadlock.

Resources in existence
$(E_1, E_2, E_3, \ldots, E_m)$

Resources available
$(A_1, A_2, A_3, \ldots, A_m)$

Current allocation matrix

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm} \end{bmatrix}$$

Request matrix

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1m} \\ R_{21} & R_{22} & R_{23} & \cdots & R_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & R_{n3} & \cdots & R_{nm} \end{bmatrix}$$

Rilevare gli stalli: esempi

- Vediamo un **esempio**:

Tape drives Plotters Scanners CD Roms

$E = (4 \quad 2 \quad 3 \quad 1)$

Tape drives Plotters Scanners CD Roms

$A = (2 \quad 1 \quad 0 \quad 0)$

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

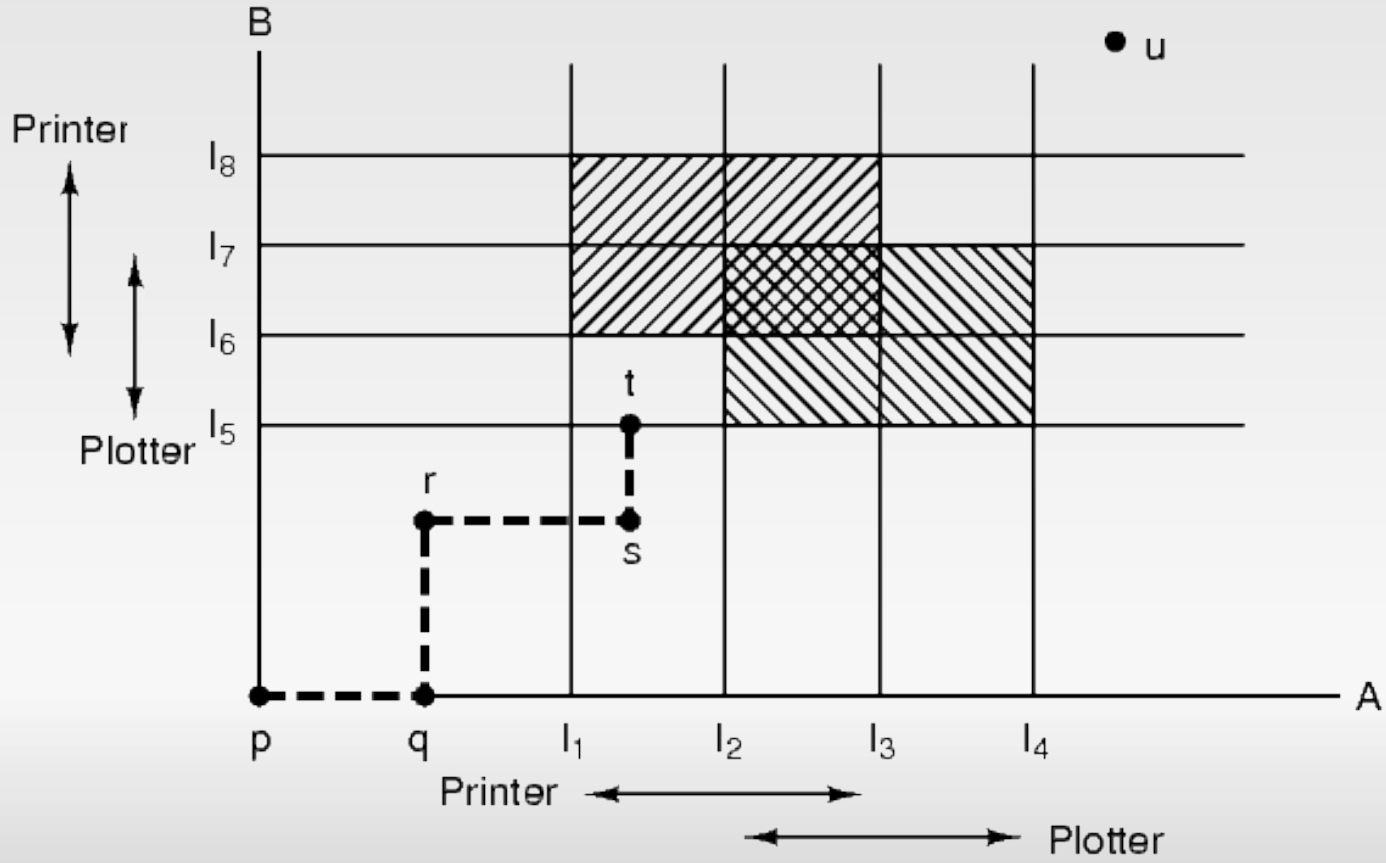
  - non abbiamo deadlock;
- proviamo una **variante**: aggiungiamo una richiesta per uno scanner al processo 3:
  - abbiamo uno stallo.

Risoluzione di uno stallo

- **Possibili approcci**:
  - **uso della preemption** su risorse (che non la supportano): può avere un costo a livello di operatività (intervento manuale o reset della risorsa);
  - **uso del rollback** ad un checkpoint precedente considerato sicuro;
  - **ripristino tramite terminazione** (con riavvio) di alcuni processi.

Evitare gli stalli

- Decidere se l'allocazione di una risorsa può essere concessa in **fase di richiesta**;
  - **bloccaggio** tramite lo scheduling e non per mancata disponibilità;
- la risorsa viene concessa se si passa ad uno **stato sicuro**: esempio.



Evitare gli stalli

- Uno stato si dice **sicuro** se esiste un possibile scheduling che permette a tutti i processi di terminare correttamente anche se questi decidono di utilizzare immediatamente le risorse al massimo;
- vediamo un **esempio** con 3 processi e una sola classe di risorse:

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3

→ B → (sicuro)

	Has	Max
A	3	9
B	4	4
C	2	7

Free: 1

(sicuro) → C → (sicuro)

	Has	Max
A	3	9
B	0	–
C	2	7

Free: 5

→ A → (sicuro)

	Has	Max
A	3	9
B	0	–
C	7	7

Free: 0

	Has	Max
A	3	9
B	0	–
C	0	–

Free: 7

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3

→ A +1 → (non sicuro)

	Has	Max
A	4	9
B	2	4
C	2	7

Free: 2

→ B →

	Has	Max
A	4	9
B	4	4
C	2	7

Free: 0

	Has	Max
A	4	9
B	—	—
C	2	7

Free: 4

STALLO!

Evitare gli stalli

- **Nota**: uno stato non sicuro non equivale ad uno stato di deadlock! Potrebbe avvenire dopo oppure non avvenire se i processi liberano alcune risorse spontaneamente prima della loro terminazione.
- l'algoritmo visto viene detto **del banchiere**;
- si applica anche al caso a **più risorse per classi**.

Process	Tape drives	Plotters	Printers	CD ROMs
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Printers	CD ROMs
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still needed

E = (6342)

A = (1020)

Prevenire gli stalli

- **Negare** una delle 4 **condizioni necessarie** già viste:
  - rimuovere la **mutua esclusione**:
    - esempio: uso di uno **spooler** per la stampante;
  - negare la condizione di **possesso e attesa**:
    - **preallocazione** in fase di avvio di tutte le risorse necessarie;
      - richiede la **conoscenza a priori** delle stesse;
      - utilizzo **non ottimale** delle risorse;
    - allocazione di risorse aggiuntive con **rilascio preventivo**;
    - possibilità di **attesa indefinita**;

Prevenire gli stalli

- negare l'**impossibilità di prelazione**;
  - applicabile solo con alcune risorse (**salvataggio dello stato**);
  - si prelazionano le risorse di:
    - l'applicazione che ha fatto una richiesta che non si può soddisfare;
    - una applicazione in attesa di altre risorse;
- negare l'**attesa circolare**;
  - **numerazione globale** delle applicazioni;
  - **vincolo** dell'acquisizione delle risorse;
  - eventuale **prerilascio** per rispettare l'ordine.