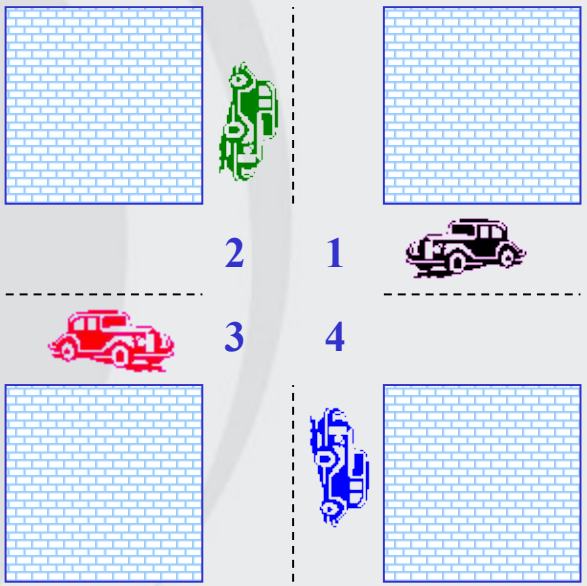
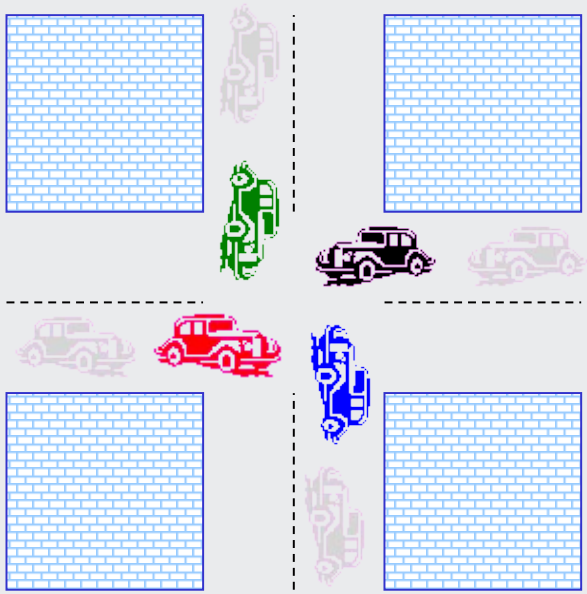
Deadlock

Principi del deadlock (stallo)
Rilevamento dello stallo
Trattamento dello stallo
Esclusione dello stallo
Starvation

Un esempio di ingorgo



Stallo possibile



Stallo

Un esempio di ingorgo

- I quadranti 1-4 sono risorse contese
  - L'automobile a destra richiede 1 e 2
  - L'automobile in alto richiede 2 e 3
  - L'automobile a sinistra richiede 3 e 4
  - L'automobile in basso richiede 4 e 1

Deadlock

- Esistono potenziali conflitti tra i processi che utilizzano risorse condivise
- La maggior parte delle risorse prevede un utilizzo esclusivo
  - Esempi: CD-ROM, plotter
- Non necessariamente i deadlock coinvolgono dispositivi di I/O: esempio database in rete

Le risorse

- I deadlock possono verificarsi quando i processi ottengono accessi esclusivi a dispositivi (o oggetti software), in generale si parlerà di **risorse**
- Le risorse possono essere di diverso tipo: dispositivi di I/O, memoria, una tupla di un database, ma vengono comunemente classificate in due tipi diversi:
  1. risorse con prerilascio: possono essere tolte ai processi senza problemi (memoria, CPU)
  2. risorse senza prerilascio: se vengono tolte ai processi si ha il fallimento dell'elaborazione (stampanti, lettori di nastro)

Le risorse

- In generale i deadlock coinvolgono risorse senza prerilascio
- La sequenza di passi necessari per l'uso di una risorsa è:
  1. Richiesta della risorsa
  2. Utilizzo della risorsa
  3. Rilascio della risorsa

     Se la risorsa non è disponibile, il processo viene fatto attendere, in alcuni sistemi il processo viene automaticamente bloccato e poi risvegliato quando la risorsa torna disponibile, in altri è il processo che deve esplicitamente gestire la situazione

Definizione di deadlock

Un insieme di processi è in deadlock se ogni processo dell'insieme è in attesa di un evento che solo un altro processo appartenente allo stesso insieme può causare

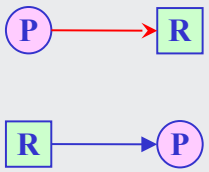
- **Un processo in deadlock non procede mai e non finisce mai la sua esecuzione, processi in deadlock possono bloccare l'intero sistema**

Condizioni necessarie

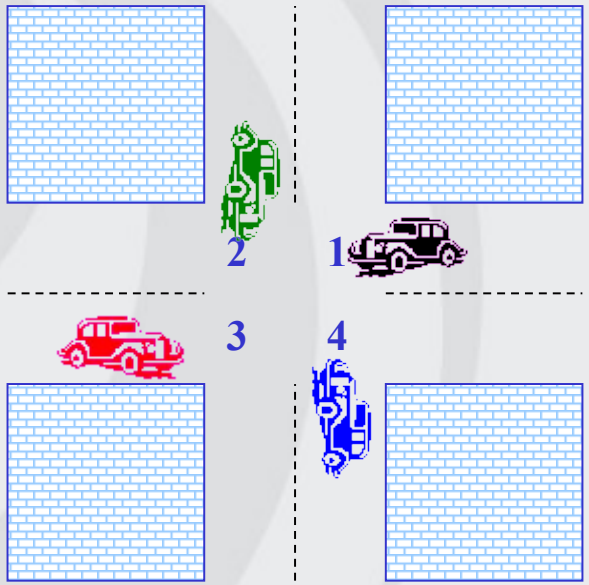
- Coffman e altri (1971) hanno dimostrato che le seguenti sono condizioni necessarie affinché esista un deadlock:
  1. Mutua esclusione: un solo processo alla volta può utilizzare la risorsa
  2. Prendi e aspetta (Hold and Wait): i processi che detengono risorse possono chiederne altre
  3. Assenza di prerilascio (No Preemption): le risorse possono essere rilasciate solo dal processo che le detiene, il rilascio non può essere forzato
  4. Attesa circolare (Circular Wait): deve esistere una lista circolare di processi/risorse

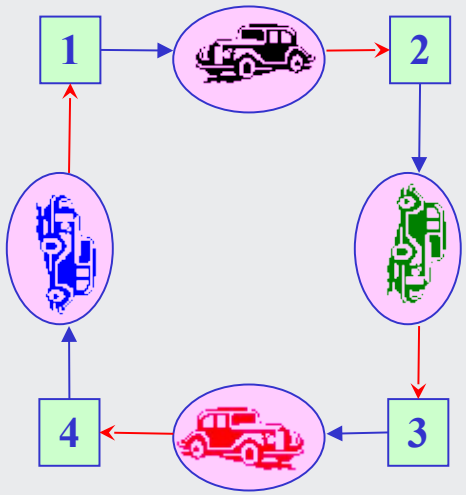
Modellazione del deadlock

- Holt (1972) ha mostrato che le quattro condizioni possono essere rappresentate da grafi orientati
- I nodi possono essere di due tipi:
  - risorse (quadrati)
  - processi (cerchi)
- Gli archi possono solo connettere nodi di tipo diverso. Necessariamente anche gli archi assumono significato diverso:
  - archi uscenti da processi: il processo ha richiesto la risorsa
  - archi uscenti da risorse: la risorsa è allocata al processo



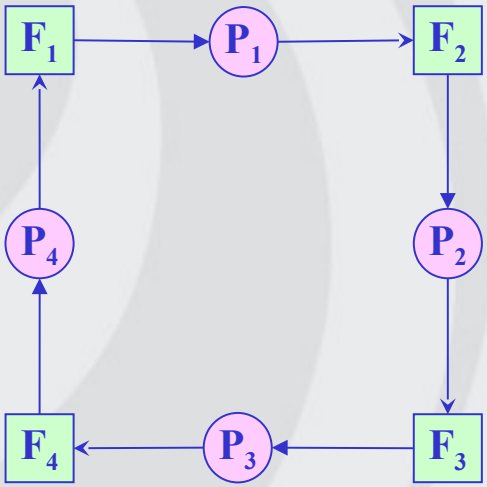
Modellazione del deadlock



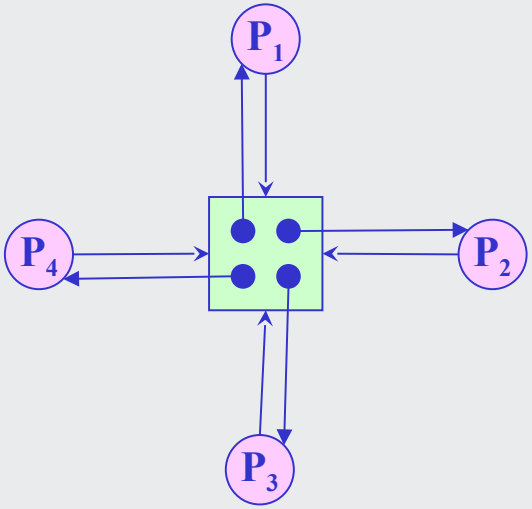


Modellazione del deadlock

- Problema dei filosofi

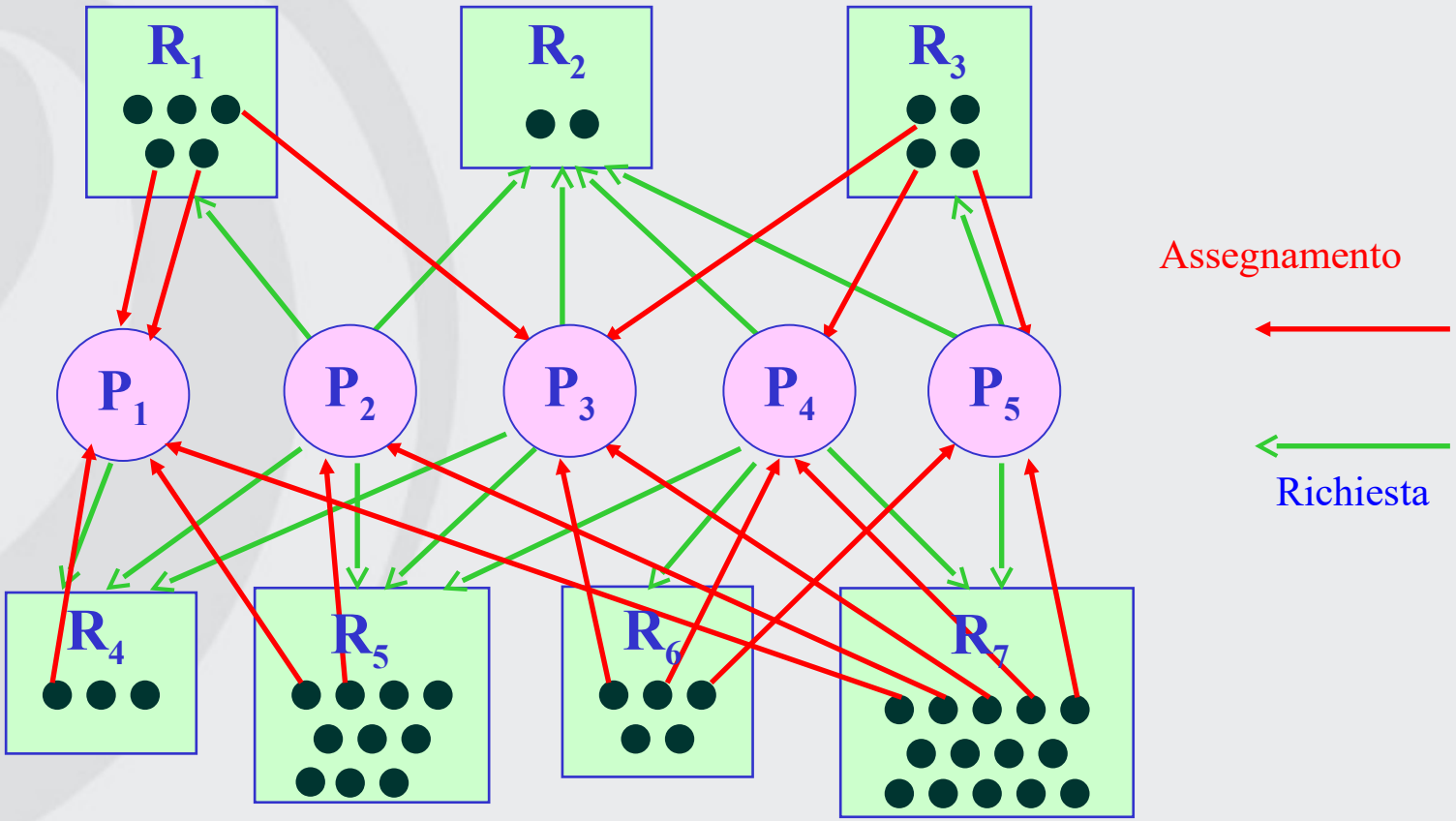


Risorse singole

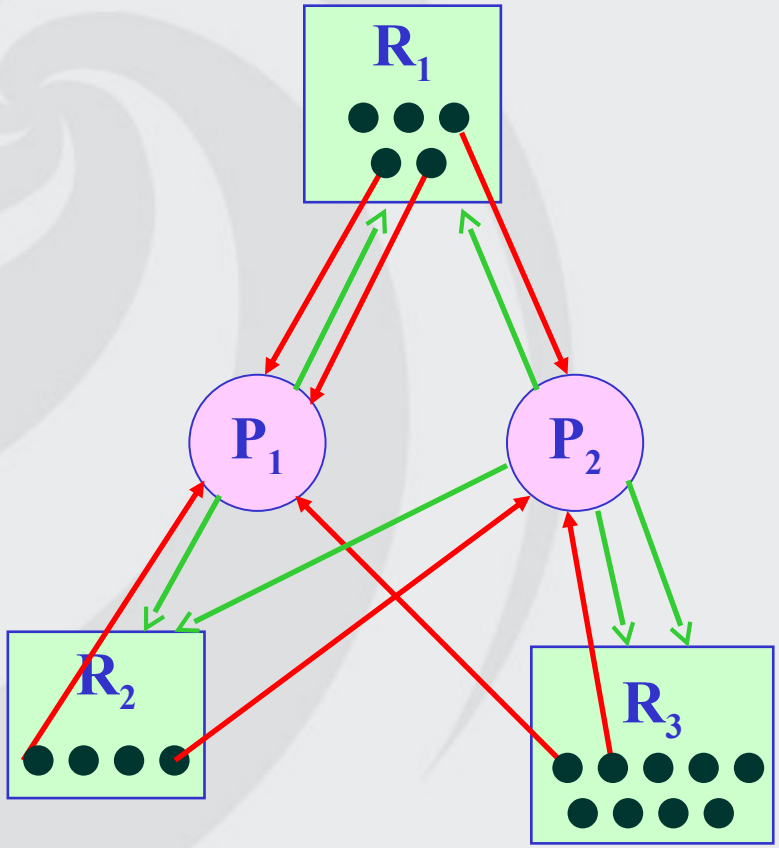


Risorse multiple

Grafo con risorse multiple

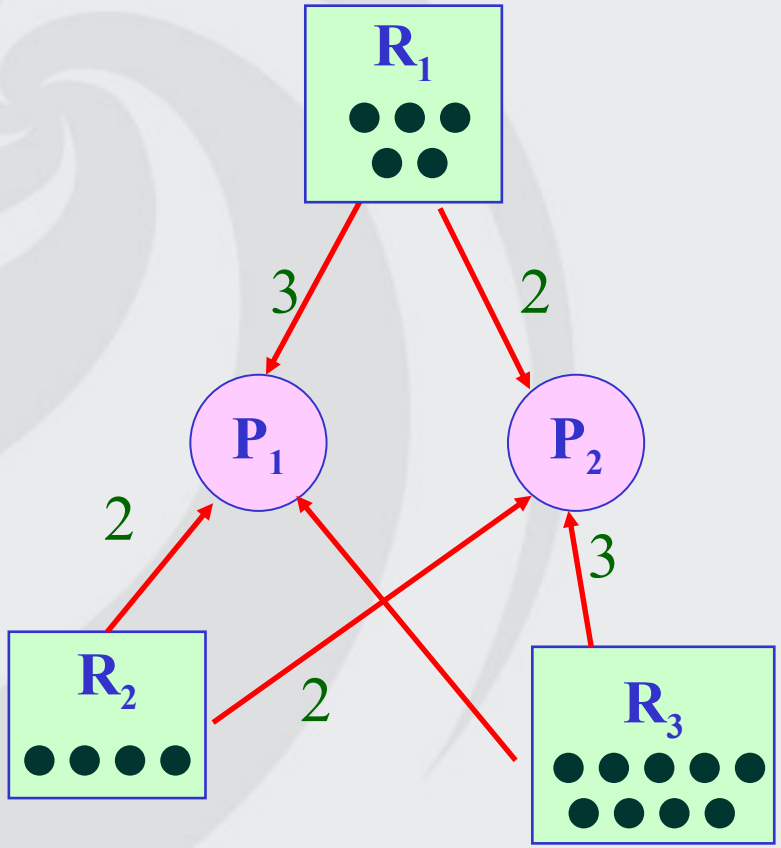


Grafo con cicli



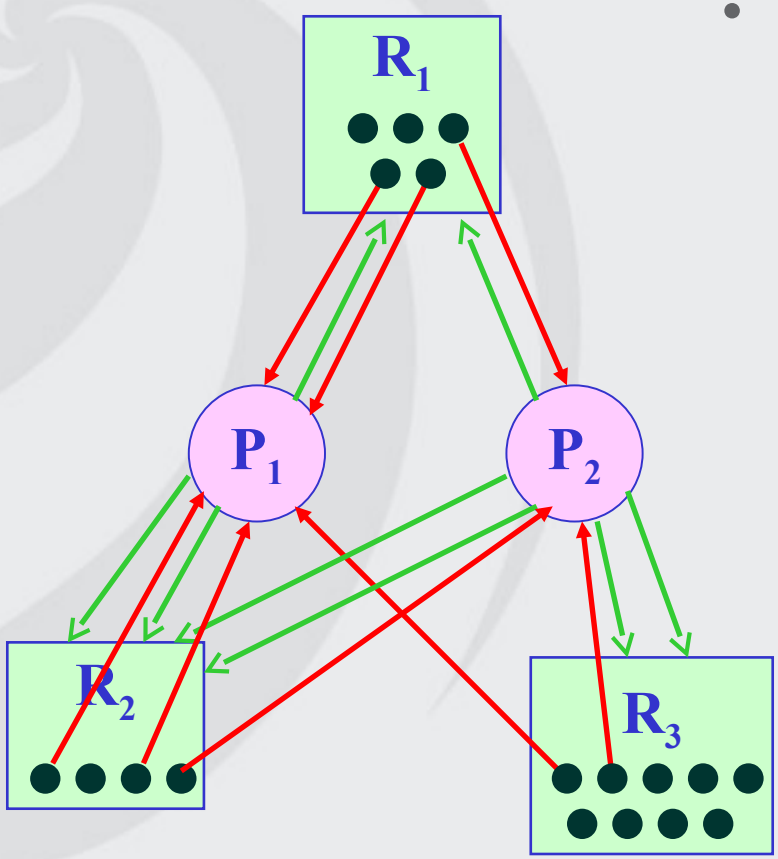
- esiste almeno un ciclo nel grafo
  - esempio di figura
    P1 → R1 → P2 → R2 → P1,
    P1 → R1 → P2 → R3 → P1
- esiste il deadlock?
  - no, esistono abbastanza risorse per tutti!

Grafo con cicli



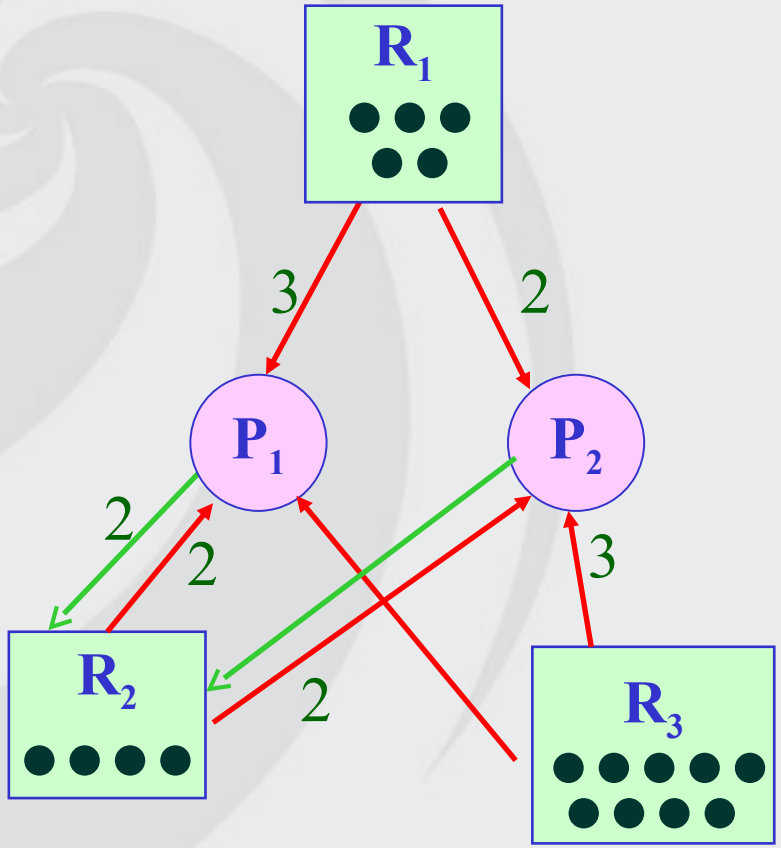
- esiste almeno un ciclo nel grafo
  - esempio di figura
    P1 → R1 → P2 → R2 → P1,
    P1 → R1 → P2 → R3 → P1
- esiste il deadlock?
  - no, esistono abbastanza risorse per tutti!

Grafo con ciclo e deadlock



- i cicli sono buoni indizi di deadlock
- È possibile il deadlock?
  - sì, in R2 non ci sono abbastanza risorse per soddisfare le richieste di P1 e P2

Grafo con cicli



- se esiste un ciclo nel grafo allora può esserci un deadlock
  - sono possibili cicli senza deadlock
  - un deadlock può verificarsi solo se esiste un ciclo

Trattamento del deadlock

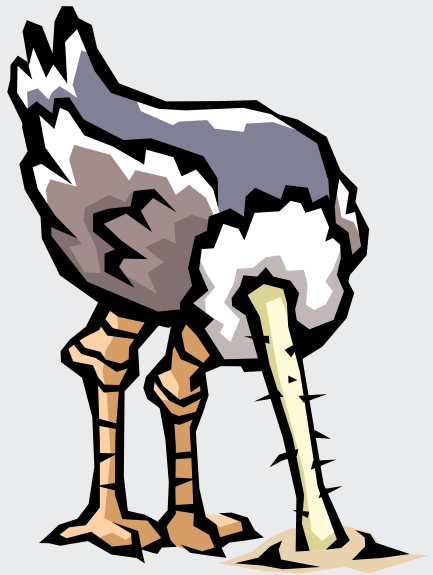
- In generale sono utilizzate quattro strategie per trattare i deadlock:
  1. Non porsi il problema
  2. Individuare il deadlock e risolverlo
  3. Prevenire il deadlock in modo dinamico
  4. Prevenire il deadlock impedendo una delle quattro condizioni necessarie

Algoritmo dello struzzo

- È l'approccio più semplice:

  far finta di niente

- Se il deadlock avviene ogni 50 anni allora è inutile preoccuparsi
- Esempio UNIX:
  - Tabella dei processi
  - Tabella dei file aperti



Prevenzione dei deadlock

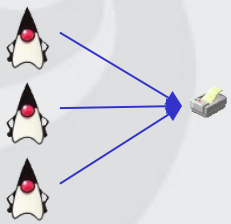
- La seconda strategia consiste nel rendere impossibile il deadlock
  - Se si evita una delle quattro condizioni necessarie il deadlock diventa impossibile

Mutua esclusione

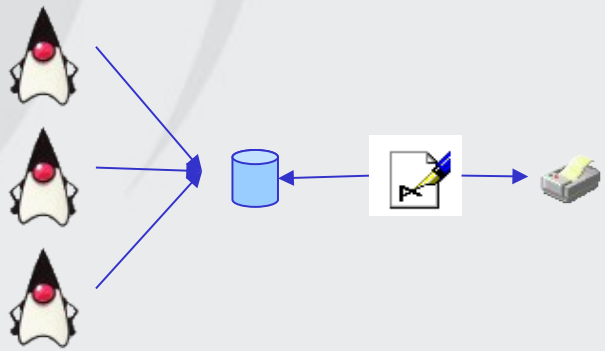
- Se non vi sono risorse assegnate in modo esclusivo ad un singolo processo non vi saranno deadlock
- Un possibile approccio è lo **Spooling**
  - Occorre un processo daemon (demone) e una directory di spooling
  - Per la stampa di un file, il processo genera l'intero file e lo memorizza nella directory di spooling
  - Il daemon è l'unico processo ad avere accesso alla directory di spooling e ha il compito di avviare la stampa
  - Lo spooling viene usato anche nei trasferimenti via rete (posta elettronica, ftp, ...)

Spooling

Esempio stampante:



- Effettuando lo spooling sulle uscite per la stampante, più processi possono produrre le loro uscite contemporaneamente
  - Dato che esiste un solo processo che utilizza la stampante fisica e non ha bisogno di altre risorse il deadlock è impossibile
  - Non tutti i problemi possono essere gestiti attraverso lo spooling (es. la tabella dei processi)
- Rimane comunque la possibilità di deadlock sulla risorsa disco



Prendi e aspetta

Occorre evitare che processi che detengono risorse rimangano in attesa di ulteriori risorse. Ci possono essere varie soluzioni:

- Un processo richiede immediatamente tutte le risorse di cui ha bisogno, se non sono disponibili attende

  Nascono nuovi problemi:

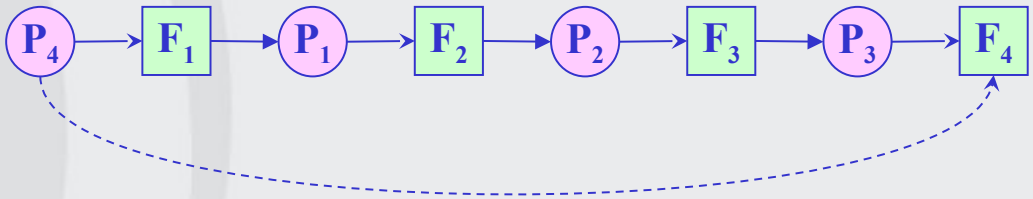
  - Un processo non sempre sa quando parte quello di cui avrà bisogno nel corso dell'esecuzione
  - L'uso delle risorse non è ottimizzato (le risorse sono spesso inutilizzate)
  - Vi è il rischio che un processo che ha bisogno di molte risorse rimanga in attesa per tempi molto lunghi
- Un modo leggermente diverso è quello di imporre che un processo rilasci temporaneamente le risorse detenute prima di una nuova richiesta

La terza condizione ***nessun prerilascio*** è di fatto inutilizzabile

Attesa Circolare

L'attesa circolare può essere eliminata in vari modi:

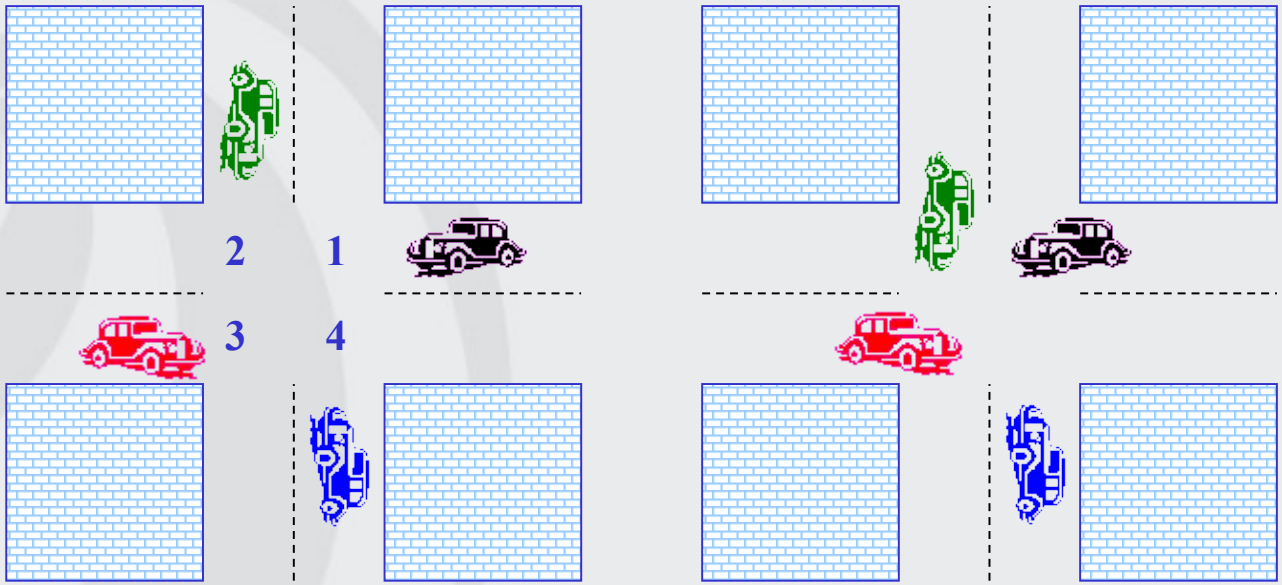
- Un processo può richiedere una sola risorsa per volta
- Le risorse sono numerate e possono essere richieste solo secondo l'ordine numerico



Il filosofo 4 non può richiedere la forchetta 4 prima della forchetta 1

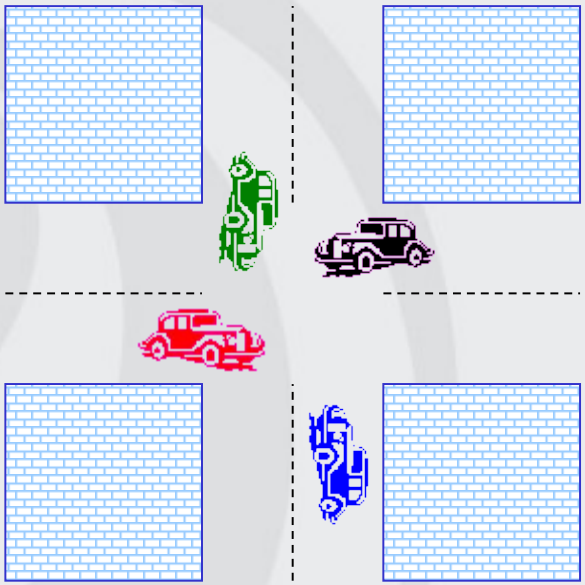
- Una variante richiede semplicemente che la nuova risorsa debba avere una etichetta maggiore di tutte quelle detenute

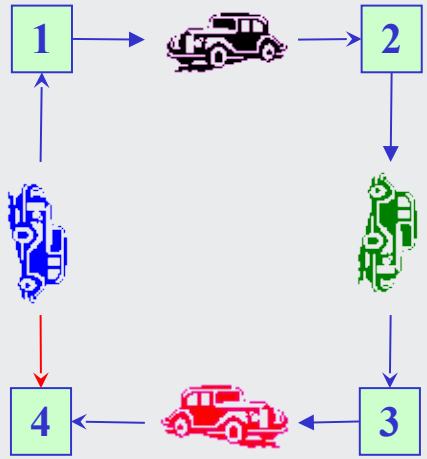
Attesa circolare



L'automobile in basso non può chiedere 4 prima di 1
Perciò l'auto a sinistra può procedere

Attesa circolare





Il grafo non descrive un ciclo

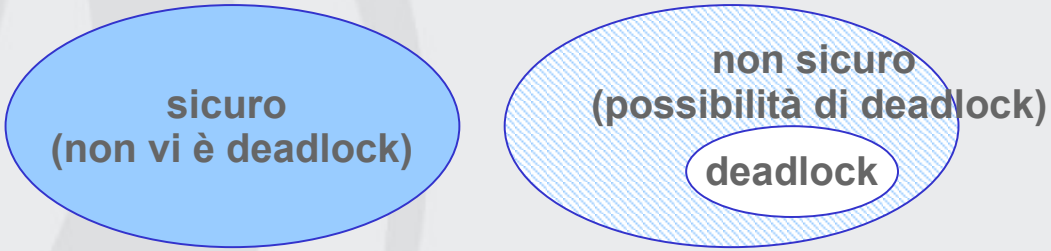
Evitare il deadlock

- È possibile impedire il verificarsi di un deadlock?

- Sì se si hanno a disposizione alcune informazioni
  - Il modello più semplice e più utile richiede che ciascun processo dichiari il numero massimo di risorse necessarie
  - L'algoritmo esamina dinamicamente lo stato di allocazione delle risorse per garantire che non accada mai una attesa circolare
  - Lo stato è definito dal numero di risorse disponibili e allocate e dal numero massimo di risorse richieste

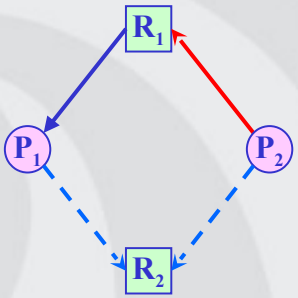
Stato sicuro

- Uno stato si dice **sicuro** se esiste una sequenza di altri stati che porta tutti i processi ad ottenere le risorse necessarie (e quindi terminare) altrimenti viene detto **non sicuro**

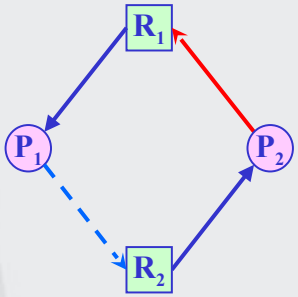


Per evitare i deadlock è quindi sufficiente evitare gli stati non sicuri

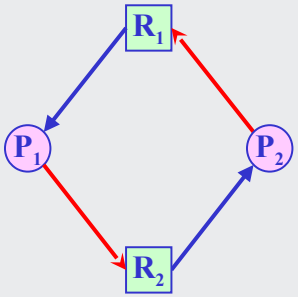
Stato sicuro/non sicuro



Stato sicuro



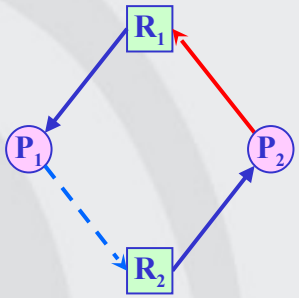
Stato non sicuro



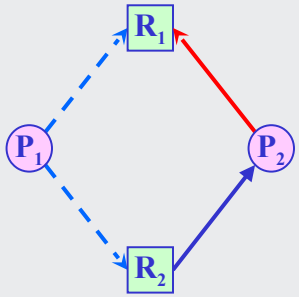
Deadlock

Possibile richiesta

Stato sicuro/non sicuro



Stato non sicuro



Stato sicuro

Da uno stato non sicuro è possibile tornare a uno stato sicuro

Algoritmo del banchiere

- Tratta risorse multiple
- Ogni processo deve dichiarare il numero massimo di risorse necessarie
- Se un processo richiede una risorsa, se non è disponibile rimane in attesa
- Dopo che un processo ha ottenuto tutte le risorse necessarie, le deve rilasciare in un tempo finito

Algoritmo del banchiere

Nome	U	M
Anna	0	6
Barbara	0	5
Carlo	0	4
Davide	0	7
Disponibili	10	

- L'algoritmo è stato proposto da Dijkstra (1965) imita il comportamento di un banchiere nei confronti delle richieste dei clienti
- Ad ogni cliente è concesso un certo numero di unità di credito (per esempio 1000 €) (cioè le risorse finanziarie Massime necessarie per portare a conclusione i rispettivi affari)
  Lo stato del sistema è la quantità di credito Usato ad ogni istante
- Complessivamente le risorse necessarie sarebbero 22 unità, ma il banchiere ne ha a disposizione solamente 10

Algoritmo del banchiere

Nome	U	M	N
Anna	1	6	5
Barbara	1	5	4
Carlo	2	4	2
Davide	4	7	3
Disp.	2		

Risorse Necessarie
N = M - U

- Uno stato si dice **sicuro** se esiste una sequenza di altri stati che porta tutti i clienti ad ottenere prestiti fino al loro massimo credito (e quindi terminare)
- È uno stato sicuro?
  - Sì perché con due unità Carlo può portare a termine i suoi affari (liberando quindi quattro unità, ...)

Algoritmo del banchiere

Nome	U	M	N
Anna	1	6	5
Barbara	2	5	3
Carlo	2	4	2
Davide	4	7	3
Disp.	1		

- Se invece di concedere credito a Carlo si concede a Barbara

- Non è uno stato sicuro perché nessun cliente riesce a portare a termine i suoi affari

Algoritmo

- Una richiesta viene evasa solo se porta in uno stato ancora sicuro altrimenti il processo deve attendere
- L'algoritmo descritto è applicabile ad un sistema con una singola risorsa multipla, ma può essere generalizzato al caso di un sistema complesso con molte classi di risorse
- L'algoritmo del banchiere *soffre* del solito problema: presuppone una conoscenza ***completa*** del sistema
- È comunque un punto di partenza per risolvere i casi concreti

Risorse multiple

- Dati: allocazione corrente, massima allocazione, risorse disponibili
  - Occorre calcolare le risorse necessarie

Processi	Corrente	Massima	Necessaria
P0	1 0 1 0 0	2 0 2 1 3	1 0 1 1 3
P1	1 1 0 1 0	1 2 1 2 2	0 1 1 1 2
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P3	0 1 1 1 1	0 1 2 1 2	0 0 1 0 1
P4	0 0 1 0 0	1 1 1 2 2	1 1 0 2 2
Disponibili	1 0 1 0 1		

L'unico processo che può ottenere tutte le risorse che gli servono è P3

Risorse multiple

Processi	Corrente	Massima	Necessaria
P0	1 0 1 0 0	2 0 2 1 3	1 0 1 1 3
P1	1 1 0 1 0	1 2 1 2 2	0 1 1 1 2
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P3	0 1 2 1 2	0 1 2 1 2	0 0 0 0 0
P4	0 0 1 0 0	1 1 1 2 2	1 1 0 2 2
Disponibili	1 0 0 0 0		

Quando un processo termina le rilascia e le risorse disponibili aumentano

Disponibili	1 1 2 1 2		

Risorse multiple

Processi	Corrente	Massima	Necessaria
P0	1 0 1 0 0	2 0 2 1 3	1 0 1 1 3
P1	1 1 0 1 0	1 2 1 2 2	0 1 1 1 2
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P3	0 0 0 0 0	0 1 2 1 2	
P4	0 0 1 0 0	1 1 1 2 2	1 1 0 2 2
Disponibili	1 1 2 1 2		

P1 può terminare

Risorse multiple

Processi	Corrente	Massima	Necessaria
P0	1 0 1 0 0	2 0 2 1 3	1 0 1 1 3
P1	0 0 0 0 0	1 2 1 2 2	
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P3	0 0 0 0 0	0 1 2 1 2	
P4	0 0 1 0 0	1 1 1 2 2	1 1 0 2 2
Disponibili	2 2 2 2 2		

P4 può terminare

Risorse multiple

Processi	Corrente	Massima	Necessaria
P0	1 0 1 0 0	2 0 2 1 3	1 0 1 1 3
P1	0 0 0 0 0	1 2 1 2 2	
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P3	0 0 0 0 0	0 1 2 1 2	
P4	0 0 0 0 0	1 1 1 2 2	
Disponibili	2 2 3 2 2		

P2 può terminare

Risorse multiple

Processi	Corrente	Massima	Necessaria
P0	1 0 1 0 0	2 0 2 1 3	1 0 1 1 3
P1	0 0 0 0 0	1 2 1 2 2	
P2	0 0 0 0 0	2 1 3 1 2	
P3	0 0 0 0 0	0 1 2 1 2	
P4	0 0 0 0 0	1 1 1 2 2	
Disponibili	2 2 3 2 3		

P0 può terminare

Disponibili	3 2 4 2 3		

Se esiste una sequenza in cui tutti i processi ottengono tutte le risorse necessarie, allora il sistema è in uno stato sicuro

Risorse multiple

Processi	Corrente	Massima	Necessaria
P0	1 0 1 0 0	2 0 2 1 3	1 0 1 1 3
P1	1 1 0 1 0	1 2 1 2 2	0 1 1 1 2
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P3	0 1 1 1 1	0 1 2 1 2	0 0 1 0 1
P4	0 0 1 0 0	1 1 1 2 2	1 1 0 2 2
Disponibili	1 0 1 0 1		

Se si concede una risorsa A a P0

Processi	Corrente	Massima	Necessaria
P0	2 0 1 0 0	2 0 2 1 3	0 0 1 1 3
P1	1 1 0 1 0	1 2 1 2 2	0 1 1 1 2
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P3	0 1 1 1 1	0 1 2 1 2	0 0 1 0 1
P4	0 0 1 0 0	1 1 1 2 2	1 1 0 2 2
Disponibili	0 0 1 0 1		

Risorse multiple

Processi	Corrente	Massima	Necessaria
P0	2 0 1 0 0	2 0 2 1 3	0 0 1 1 3
P1	1 1 0 1 0	1 2 1 2 2	0 1 1 1 2
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P4	0 0 1 0 0	1 1 1 2 2	1 1 0 2 2
Disponibili	0 1 2 1 2		

Processi	Corrente	Massima	Necessaria
P0	2 0 1 0 0	2 0 2 1 3	0 0 1 1 3
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
P4	0 0 1 0 0	1 1 1 2 2	1 1 0 2 2
Disponibili	1 2 2 2 2		

Risorse multiple

Processi	Corrente	Massima	Necessaria
P0	2 0 1 0 0	2 0 2 1 3	0 0 1 1 3
P2	0 0 0 0 1	2 1 3 1 2	2 1 3 1 1
Disponibili	1 2 3 2 2		

Non è più possibile portare a termine nessuno dei due processi

Rilevamento del deadlock

- Si ammette che il sistema possa entrare in uno stato di deadlock
- Richiede un algoritmo di rilevamento
- Occorre definire uno schema di recovery

Individuare lo stallo e risolverlo

- Periodicamente si controlla il grafo delle risorse per verificare l'esistenza di un ciclo, se esiste si elimina (almeno) uno dei processi coinvolti
- Un metodo più semplice non controlla nemmeno il grafo: se un processo è bloccato da troppo tempo (es. un'ora) viene rimosso
- Esiste comunque il pericolo di effetti collaterali (non sempre è possibile annullare tutti gli effetti di un processo non concluso positivamente)

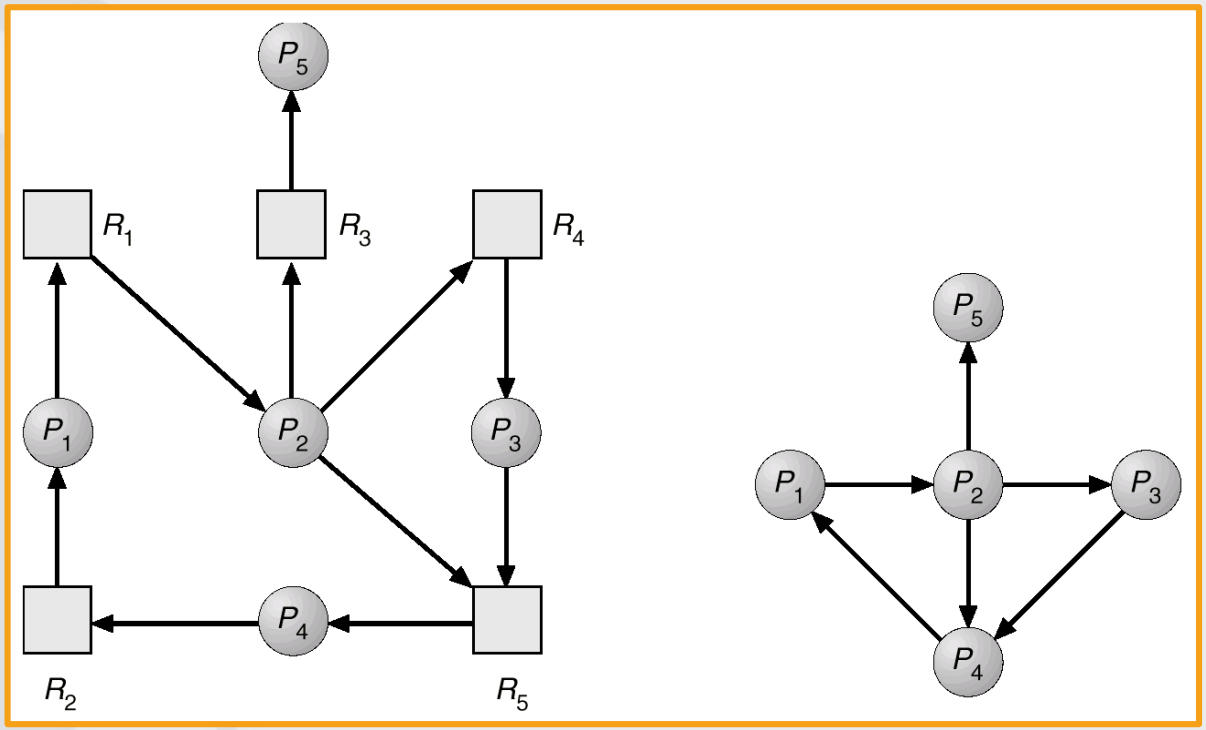
Rilevamento a singola istanza

- tutti i tipi di risorse possono avere una sola istanza
- allora un ciclo nel grafo delle allocazioni indica un deadlock
- un generico algoritmo di rilevamento richiede una complessità dell'ordine di

$$n^2 \text{ operazioni}$$

ove n è il numero di processi del grafo di attesa

Grafo di attesa e di allocazione



Grafo di allocazione

Corrispondente grafo di attesa

Rilevamento a istanze multiple

- esistono più istanze per ogni tipo di risorsa
- si usa un algoritmo simile a quello del banchiere per rilevare il deadlock
  - richiede una complessità dell'ordine di
  - $m * n^2$ operazioni
    (m = numero di tipi di risorse, n = numero di processi)
- la frequenza con cui si attiva l'algoritmo di rilevamento dipende
  - dalla probabilità che si verifichi un deadlock
  - dal numero di processi coinvolti

Strategie di recovery

- Dopo il rilevamento di un deadlock che fare?
  - preemption
  - rollback
  - terminazione di processi

Terminazione di processi

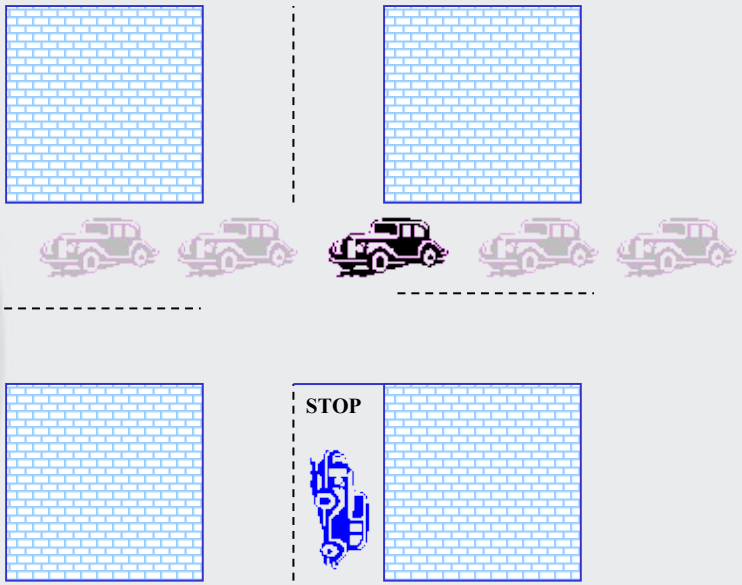
- Uccidere tutti i processi coinvolti
- Uccidere solo un processo alla volta fino a che il ciclo di deadlock non si è aperto
- Problema: quale ordine scegliere:
  - Priorità del processo
  - Tempo di esecuzione trascorso e/o necessario
  - Risorse usate/necessarie
  - Numero di processi che devono essere terminati
  - Processi interattivi o batch

Starvation

Un fenomeno simile al deadlock è la starvation

- un processo non può procedere perché altri processi detengono sempre una risorsa di cui ha bisogno
  - la richiesta del processo non viene mai soddisfatta
- in linea di principio, il processo potrebbe ottenere la risorsa, ma non l'ottiene perché:
  - è a priorità bassa
  - la temporizzazione della richiesta della risorsa è errata
  - l'algoritmo per l'allocazione delle risorse è costruito male

Starvation



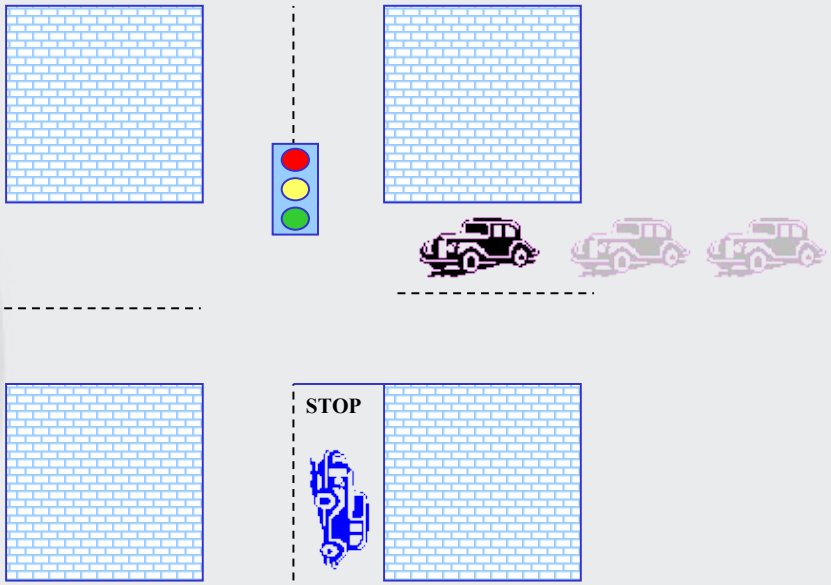
L’auto allo stop attende all’infinito, perché le auto della strada principale hanno una priorità più alta

Starvation

Soluzioni:

- opportunità: ogni processo ha una percentuale adeguata di tutte le risorse
- tempo: la priorità del processo viene aumentata se il processo aspetta per un tempo troppo lungo

Starvation



L'introduzione di un semaforo può risolvere il problema

Starvation e priorità

Problema: 2 processi uno ad alta priorità H e uno a bassa L con dati condivisi

1°) approccio (codice semplificato di H):

```
while(condizione);
```

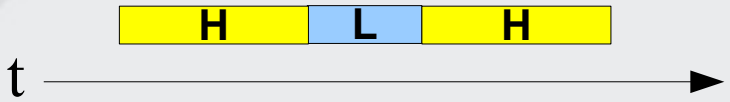
H è bloccato in un ciclo infinito, L non otterrà mai la CPU

2°) approccio introduzione di un semaforo (o altri meccanismi di sincronizzazione):

```
condizione.down();
```

H si sospende, L può lavorare (e cambiare la condizione)

H a questo punto riprende



Starvation e priorità

Problema: vi è un terzo processo M a priorità intermedia

- Va in esecuzione M (non L)

- H è bloccato per sempre (o almeno fino a che M non abbia finito) - starvation
  - H viene bloccato da un processo a priorità più bassa (problema dell'inversione di priorità)
- Possibile soluzione: cessione di priorità, H passa la sua priorità a L, che quindi sarà immediatamente eseguito, poi L restituisce la priorità a H

CVMLab
Computer Vision
& Multimedia Lab

Algoritmo del banchiere

Esercizi

Università
degli Studi
di Pavia

Esercizio

Processo	U	M
A	2	6
B	2	3
C	2	6
D	2	6
Disp.	1	

Se il sistema ha complessivamente 9 risorse, quale(/i) affermazione(/i) è(/sono) corretta(/e)?

1) Il sistema è in deadlock
2) Il sistema è in starvation
3) Il sistema è in uno stato sicuro
4) Il sistema non è in uno stato sicuro

MOTIVARE LA RISPOSTA!!!

B può terminare, ma poi con 3 risorse nessun altro processo può finire
Senza altre informazioni non è detto che il sistema sia in deadlock

Esercizio 1 – un unico tipo di risorse

Processo	U	M
A	3	7
B	2	5
C	1	6
D	2	9
Totali	8	

Dire quali richieste sono accettabili da parte del sistema con 12 o 14 risorse a disposizione:
(A,2) (B,2) (C,3) (D,4)

Esercizio 1 – dopo (A, 2)

Processo	U	M
A	5	7
B	2	5
C	1	6
D	2	9
Totali	10	

Con 12 risorse a disposizione posso concludere il processo A …

Esercizio 1 – dopo (B, 2)

Processo	U	M
A	3	7
B	4	5
C	1	6
D	2	9
Totali	10	

Con 12 risorse a disposizione posso concludere il processo B …

Esercizio 1 – dopo (C, 3)

Processo	U	M
A	3	7
B	2	5
C	4	6
D	2	9
Totali	11	

Con 12 risorse a disposizione non posso concludere alcun processo
Con 14 il processo B …

Esercizio 1 – dopo (D, 4)

Processo	U	M
A	3	7
B	2	5
C	1	6
D	6	9
Totali	12	

Con 12 o 14 risorse a disposizione non posso concludere alcun processo

Esercizio 2 – diversi tipi di risorse

Un sistema ha quattro processi e cinque risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Corrente	Massima
A	1 0 2 1 1	1 1 2 1 2
B	2 0 1 1 0	2 2 2 1 0
C	1 1 0 1 0	2 1 5 1 0
D	1 1 1 1 0	1 1 2 2 1
Disponibili	0 0 x 1 1	

Quale è il più piccolo valore di x perché questo stato sia sicuro?

Esercizio 2 – x=1 (a)

Processi	Corrente	Massima	Necessaria
A	1 0 2 1 1	1 1 2 1 2	0 1 0 0 1
B	2 0 1 1 0	2 2 2 1 0	0 2 1 0 0
C	1 1 0 1 0	2 1 5 1 0	1 0 5 0 0
D	1 1 1 1 0	1 1 2 2 1	0 0 1 1 1
Disponibili	0 0 x 1 1		

Esercizio 2 – x=1 (b)

Processi	Corrente	Massima	Necessaria
A	1 0 2 1 1	1 1 2 1 2	0 1 0 0 1
B	2 0 1 1 0	2 2 2 1 0	0 2 1 0 0
C	1 1 0 1 0	2 1 5 1 0	1 0 5 0 0
D	1 1 1 1 0	1 1 2 2 1	0 0 1 1 1
Disponibili	0 0 1 1 1		

Esercizio 2 – x=1 (c)

Processi	Corrente	Massima	Necessaria
A	1 0 2 1 1	1 1 2 1 2	0 1 0 0 1
B	2 0 1 1 0	2 2 2 1 0	0 2 1 0 0
C	1 1 0 1 0	2 1 5 1 0	1 0 5 0 0
Disponibili	1 1 2 2 1		

Esercizio 2 – x=1 (d)

Processi	Corrente	Massima	Necessaria
B	2 0 1 1 0	2 2 2 1 0	0 2 1 0 0
C	1 1 0 1 0	2 1 5 1 0	1 0 5 0 0
Disponibili	2 1 4 3 2		

Una risorsa di tipo 3 non basta, ne servono almeno 2 (x=2)

Esercizio 2 – x=2

Con x=2 (i passi precedenti sono ovviamente uguali)

Processi	Corrente	Massima	Necessaria
B	2 0 1 1 0	2 2 2 1 0	0 2 1 0 0
C	1 1 0 1 0	2 1 5 1 0	1 0 5 0 0
Disponibili	2 1 5 3 2		

Processi	Corrente	Massima	Necessaria
B	2 0 1 1 0	2 2 2 1 0	0 2 1 0 0
Disponibili	3 2 5 4 2		

Esercizio 3

Un sistema ha 4 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime	Necessarie
A	1 1 1 1	2 1 2 1	1 0 1 0
B	2 0 1 0	2 4 3 2	0 4 2 2
C	2 0 2 2	5 4 2 2	3 4 0 0
D	0 2 1 1	0 3 4 1	0 1 3 0

Disponibili
1 1 3 1

Richiesta B
0 1 1 0

Esercizio 3

Un sistema ha 4 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime	Necessarie
A	1 1 1 1	2 1 2 1	1 0 1 0
B	2 1 2 0	2 4 3 2	0 3 1 2
C	2 0 2 2	5 4 2 2	3 4 0 0
D	0 2 1 1	0 3 4 1	0 1 3 0

Disponibili
1 0 2 1

Il processo A può terminare

Esercizio 3

Un sistema ha 4 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime	Necessarie
B	2 1 2 0	2 4 3 2	0 3 1 2
C	2 0 2 2	5 4 2 2	3 4 0 0
D	0 2 1 1	0 3 4 1	0 1 3 0

Disponibili
2 1 3 2

Il processo D può terminare

Esercizio 3

Un sistema ha 4 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime	Necessarie
B	2 1 2 0	2 4 3 2	0 3 1 2
C	2 0 2 2	5 4 2 2	3 4 0 0

Disponibili
2 3 4 3

Il processo B può terminare

Processi	Allocate	Massime	Necessarie
C	2 0 2 2	5 4 2 2	3 4 0 0

Disponibili
4 4 6 3

Il processo C può terminare

Esercizio 4

Un sistema ha 5 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime
A	4 X 3 2	6 4 5 6
B	8 0 Y 2	10 7 6 8
C	4 0 0 0	6 2 0 8
D	0 0 3 2	0 3 4 2
E	2 1 Z 4	9 1 6 9

Disponibili
2 2 10 4

Esercizio 4

Un sistema ha 5 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime	Necessarie
A	4 X 3 2	6 4 5 6	2 4-X 2 4
B	8 0 Y 2	10 7 6 8	2 7 6-Y 6
C	4 0 0 0	6 2 0 8	2 2 0 8
D	0 0 3 2	0 3 4 2	0 3 1 0
E	2 1 Z 4	9 1 6 9	7 0 6-Z 5

Disponibili
2 2 10 4

Solo A può terminare purché $4 \geq X \geq 2$

Esercizio 4

Un sistema ha 5 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime	Necessarie
B	8 0 Y 2	10 7 6 8	2 7 6-Y 6
C	4 0 0 0	6 2 0 8	2 2 0 8
D	0 0 3 2	0 3 4 2	0 3 1 0
E	2 1 Z 4	9 1 6 9	7 0 6-Z 5

Disponibili
6 2+X 13 6

D può terminare

Esercizio 4

Un sistema ha 5 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime	Necessarie
B	8 0 Y 2	10 7 6 8	2 7 6-Y 6
C	4 0 0 0	6 2 0 8	2 2 0 8
E	2 1 Z 4	9 1 6 9	7 0 6-Z 5

Disponibili
6 2+X 16 8

C può terminare

Esercizio 4

Un sistema ha 5 processi e 4 risorse allocabili. L'allocazione corrente e il massimo fabbisogno sono i seguenti:

Processi	Allocate	Massime	Necessarie
B	8 0 Y 2	10 7 6 8	2 7 6-Y 6
E	2 1 Z 4	9 1 6 9	7 0 6-Z 5

Disponibili
10 2+X 16 8

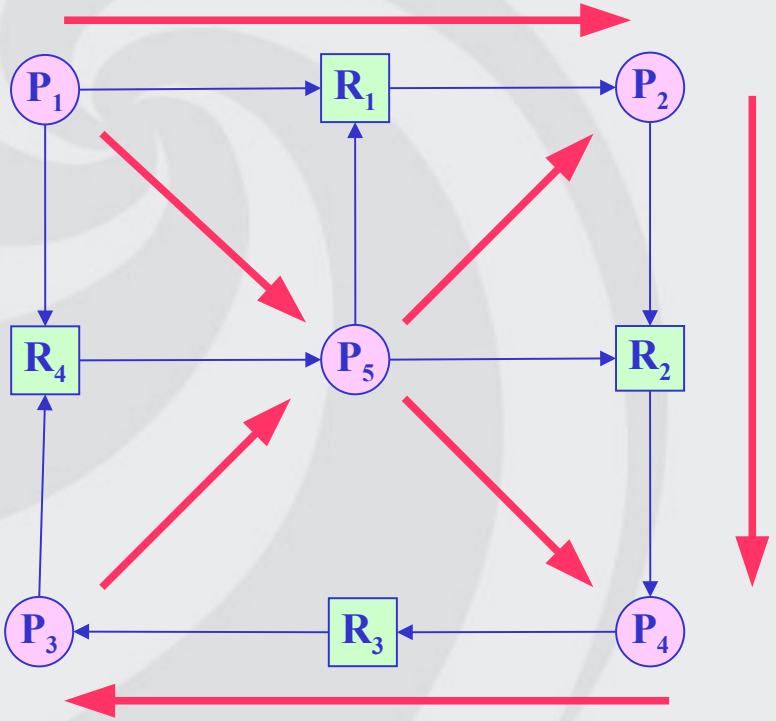
E può terminare $6 \geq Z \geq 0$

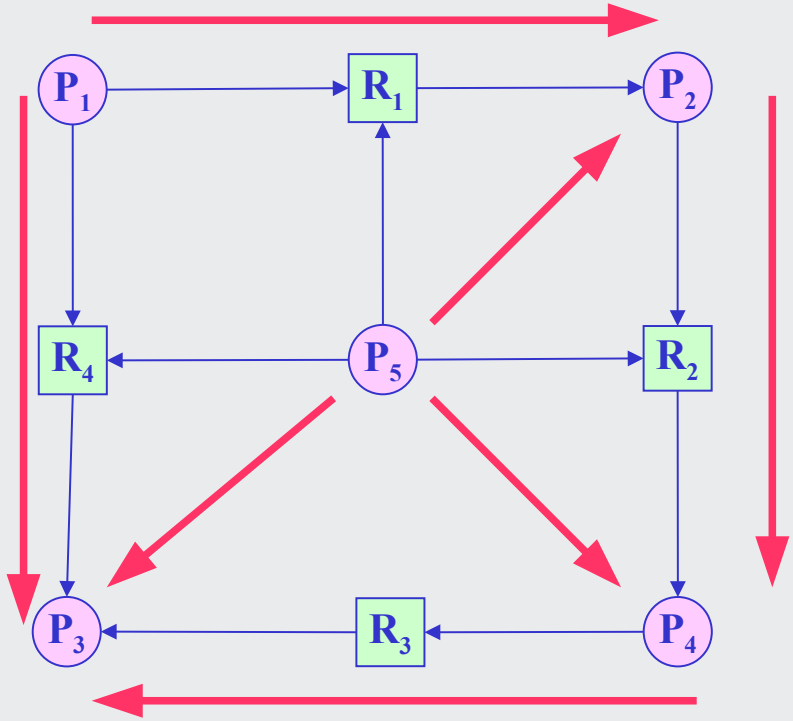
Processi	Allocate	Massime	Necessarie
B	8 0 Y 2	10 7 6 8	2 7 6-Y 6

Disponibili
12 3+X 16+Z 12

B può terminare se $X=4$ e $6 \geq Y \geq 0$

Grafo di allocazione e grafo di attesa





Algoritmo del banchiere

- Un sistema ha 3 processi e 4 stampanti identiche. Ogni processo richiede al massimo 2 stampanti.
  - Può accadere uno stallo?
  - Se ci sono M processi che richiedono al massimo N risorse, quante risorse dovrebbero essere presenti nel sistema per avere la garanzia che non si verifichi mai lo stallo?
  - Cosa cambia se le risorse vengono sempre allocate a coppie?

Algoritmo del banchiere

Se il sistema è in uno stato non sicuro è anche in stallo?

Processo	Risorse Assegnate	Necessità massima
P1	1	3
P2	1	3
Disponibili	1	

Stato non sicuro

Passo 1: P1 rilascia una risorsa
Passo 2: P2 ottiene 2 risorse e termina
Passo 3: P1 ottiene 3 risorse e termina

Passo 1: P1 ottiene una risorsa
Passo 2: P2 richiede una risorsa (attende)
Passo 3: P1 richiede una risorsa (attende) Il sistema è in stallo

Nessuna verifica del deadlock

In un sistema vi sono 2 processi (A, B) che per terminare necessitano di 3 risorse (indistinguibili).
Le risorse effettivamente disponibili sono 4.
La sequenza delle richieste è: A, B, A, B, A, B.
Mostrare l'evoluzione del sistema se non viene gestito deadlock oppure se si utilizza l'algoritmo del banchiere

Stato iniziale

Dopo le prime 4 richieste

Dopo la 5ª richiesta

Dopo la 6ª richiesta
Il sistema è in deadlock

Uso dell'algoritmo del banchiere

Stato iniziale

Dopo le prime 3 richieste

Verifica 4ª richiesta
Non è uno stato sicuro
La richiesta non viene accolta
B rimane in sospeso

Dopo la 5ª richiesta
(accolta – lo stato è
ovviamente sicuro)

Il processo A termina
Le richieste di B possono
ora essere considerate

Il processo B termina

Algoritmo del banchiere

Processi	Prestito attuale				Necessità massima				Richiesta attuale			
	A	B	C	D	A	B	C	D	A	B	C	D
1	1	0	2	0	3	2	4	2	2	2	2	2
2	0	3	1	2	3	5	1	2	3	2	0	0
3	2	4	5	1	2	7	7	5	0	3	2	4
4	3	0	0	6	5	5	0	8	2	5	0	2
5	4	2	1	3	6	2	1	4	2	0	0	1
Disponibili	3	4	0	1								
Totali	13	13	9	13								

Algoritmo del banchiere

Processi	Prestito attuale				Necessità massima				Richiesta attuale			
	A	B	C	D	A	B	C	D	A	B	C	D
1	1	0	2	0	3	2	4	2	2	2	2	2
2	0	3	1	2	3	5	1	2	3	2	0	0
3	2	4	5	1	2	7	7	5	0	3	2	4
4	3	0	0	6	5	5	0	8	2	5	0	2
5	4	2	1	3	6	2	1	4	2	0	0	1
Disponibili	3	4	0	1								
Totali	13	13	9	13								

Algoritmo del banchiere

Processi	Prestito attuale				Necessità massima				Richiesta attuale			
	A	B	C	D	A	B	C	D	A	B	C	D
1	1	0	2	0	3	2	4	2	2	2	2	2
2	0	3	1	2	3	5	1	2	3	2	0	0
3	2	4	5	1	2	7	7	5	0	3	2	4
4	3	0	0	6	5	5	0	8	2	5	0	2
Disponibili	7	6	1	4								
Totali	13	13	9	13								

Algoritmo del banchiere

Processi	Prestito attuale				Necessità massima				Richiesta attuale			
	A	B	C	D	A	B	C	D	A	B	C	D
1	1	0	2	0	3	2	4	2	2	2	2	2
2	0	3	1	2	3	5	1	2	3	2	0	0
3	2	4	5	1	2	7	7	5	0	3	2	4
Disponibili	10	6	1	10								
Totali	13	13	9	13								

Processi	Prestito attuale				Necessità massima				Richiesta attuale			
	A	B	C	D	A	B	C	D	A	B	C	D
1	1	0	2	0	3	2	4	2	2	2	2	2
3	2	4	5	1	2	7	7	5	0	3	2	4
Disponibili	10	9	2	12								
Totali	13	13	9	13								

Algoritmo del banchiere

- Supponiamo ci sia un deadlock
  - È necessario che tutti i processi coinvolti appartengano a un percorso chiuso?