Leggete il diario!

http://matteo.vaccari.name/so/diario.php

Dopo ogni lezione trovate:
- gli argomenti
- le sezioni del testo corrispondenti
- i lucidi
- documentazione addizionale
- esercizi

Tutte le letture riportate nel diario **sono parte del programma di esame**, a meno che non siano esplicitamente dichiarate *facoltative*

Deadlock

Processi: accesso esclusivo alle risorse

Due tipi di risorse:
- prelazionabili (preemptable)
  il processo può sopportare la sottrazione della risorsa
- non prelazionabili
  il processo cade se la risorsa è sottratta

Risorse

Ciascun processo:
0. richiede la risorsa
1. la usa
2. la rilascia

Se la risorsa non è disponibile, il processo
- resta bloccato, oppure
- riceve un codice di errore

Definizione di deadlock

A set of processes is *deadlocked* if each process in the set is waiting for an event that only another process in the set can cause

Condizioni necessarie per un deadlock

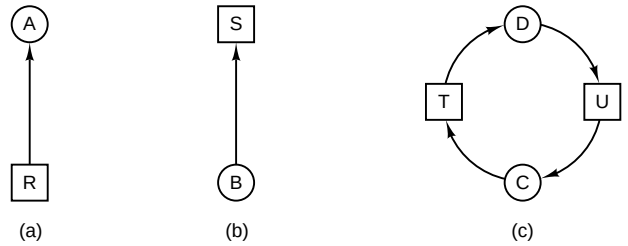
0. Mutua esclusione
   risorsa assegnata a *un* processo oppure disponibile

1. Hold and wait
   un processo ha una risorsa e ne può chiedere altre

2. Senza prelazione
   le risorse possono essere restituite solo volontariamente

3. Attesa circolare
   ci deve essere una catena di processi

Modellazione dei deadlock con grafi



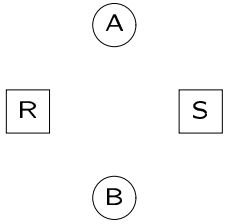
(a) il processo *A* ha assegnata la risorsa *R*

(b) il processo *B* chiede la risorsa *S*

(c) deadlock

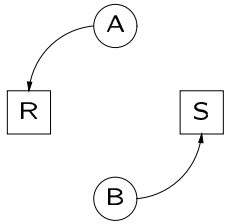
Come avviene un deadlock (i)

Processo A	**Processo B**
richiedi R	richiedi S
richiedi S	richiedi R
usa R e S	usa R e S
rilascia S	rilascia S
rilascia R	rilascia R



Come avviene un deadlock (ii)

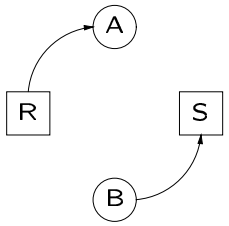
Processo A	**Processo B**
•richiedi R	•richiedi S
richiedi S	richiedi R
usa R e S	usa R e S
rilascia S	rilascia S
rilascia R	rilascia R



Il sistema operativo deve decidere quale risorsa assegnare per prima

Come avviene un deadlock (iii)

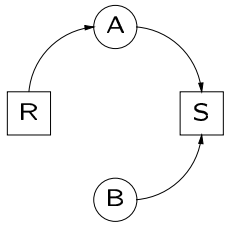
Processo A	**Processo B**
•richiedi R	•richiedi S
richiedi S	richiedi R
usa R e S	usa R e S
rilascia S	rilascia S
rilascia R	rilascia R



Il sistema operativo ha accordato la richiesta di *A*

Come avviene un deadlock (iv)

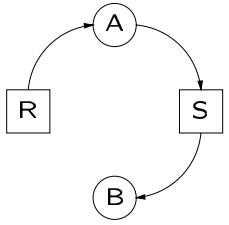
Processo A	**Processo B**
richiedi R	•richiedi S
•richiedi S	richiedi R
usa R e S	usa R e S
rilascia S	rilascia S
rilascia R	rilascia R



A prosegue e richiede la risorsa *S*

Come avviene un deadlock (v)

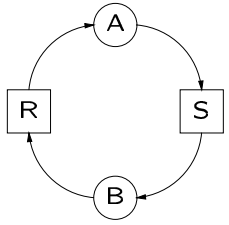
Processo A	**Processo B**
richiedi R	•richiedi S
•richiedi S	richiedi R
usa R e S	usa R e S
rilascia S	rilascia S
rilascia R	rilascia R



Il sistema operativo ha accordato la richiesta di *B*
brutta mossa!

Come avviene un deadlock (vi)

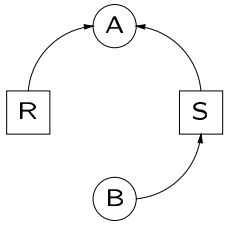
Processo A	**Processo B**
richiedi R	richiedi S
•richiedi S	•richiedi R
usa R e S	usa R e S
rilascia S	rilascia S
rilascia R	rilascia R



L'unico esito possibile ora è il deadlock

Potevamo evitarlo al passo v

Processo A	**Processo B**
richiedi R	•richiedi S
richiedi S	richiedi R
•usa R e S	usa R e S
rilascia S	rilascia S
rilascia R	rilascia R



Il sistema operativo accorda l'altra richiesta di *A*

Strategie per gestire i deadlock

0. ignorare il problema (!)

1. detection and recovery

2. algoritmi di allocazione risorse "prudenti"

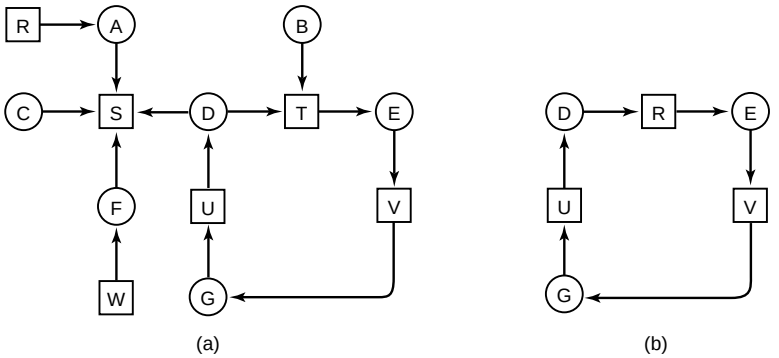
3. prevenzione (neghiamo una delle 4 condizioni necessarie)

0. L'algoritmo dello struzzo

Facciamo finta che non ci sia problema

Ragionevole se:
- il deadlock è molto improbabile
- il costo della prevenzione è molto alto

1. Detection and recovery: detection



Tenere traccia di allocazioni e richieste

Trovare i deadlock è un semplice problema di teoria dei grafi

1. Detection and recovery: recovery

Abbiamo trovato un deadlock; e ora? Tre possibilità

Preemption
togliamo la risorsa; il processo riceve un codice di errore

Rollback
lo stato del processo viene fatto "retrocedere" a prima della acquisizione

Termination
il processo viene ucciso

2. Algoritmi specializzati

Algoritmi per allocazione di risorse

Non cedere mai una risorsa se questo comporta la possibilità di un deadlock

Questi algoritmi dipendono da assunzioni sul comportamento futuro dei processi

3. Prevenzione

Neghiamo una delle 4 condizioni necessarie

0. Mutua esclusione
risorsa assegnata a *un* processo oppure disponibile

Alcune risorse possono essere rese condivisibili

Es. spooling; code di eventi

Non tutte le risorse si prestano a questa soluzione

3. Prevenzione

neghiamo
1. Hold and wait
un processo ha una risorsa e ne può chiedere altre, una per volta

tutte le risorse sono acquisite alla partenza di un processo (es. sistemi batch)
- inefficiente
- ma efficace

3. Prevenzione

neghiamo
2. Senza prelazione
   le risorse possono essere restituite solo volontariamente

- poco pratico

3. Prevenzione

neghiamo
3. Attesa circolare
   ci deve essere una catena di processi

idea: imponiamo un ordine sulle risorse

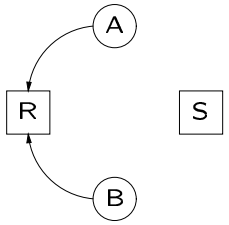
i processi devono richiedere le risorse nell'ordine

impossibile creare un circolo

es. il kernel di Linux

Le richieste vanno fatte in ordine; supponiamo $R < S$

Processo A	**Processo B**
•richiedi R	•richiedi R
richiedi S	richiedi S
usa R e S	usa R e S
rilascia S	rilascia S
rilascia R	rilascia R



A e B competono per R; solo uno dei due potrà proseguire