Sistemi Operativi A
Esercizi

Ombretta Gaggi
Università Ca' Foscari Venezia
Corso di Laurea in Informatica

Esercizio 1 – Gestione Risorse

- Quattro Processi, P1-P4, richiedono 4 risorse R1-R4. Si supponga che le risorse vengano allocate ad un ad una e che nessuna delle risorse venga rilasciata da un processo prima della sua terminazione. In un certo istante la fotografia del sistema è rappresentata dalle tabelle:

risorse totali

R1	R2	R3	R4
4	6	5	7

allocazione

R1	R2	R3	R4
1	0	2	2
1	2	0	1
2	1	2	0
0	0	0	2

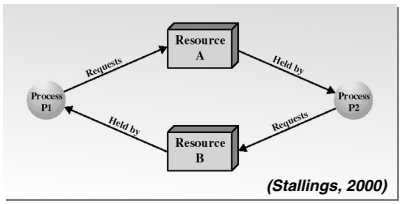
richieste

R1	R2	R3	R4
0	1	0	2
0	4	1	0
0	2	0	5
1	2	1	0

- Può verificarsi un *deadlock*?
- Se sì, esiste una strategia di allocazione che evita il deadlock?

Richiami - Deadlock

- Perché si verifichi un *deadlock* occorrono quattro condizioni
  - *mutua esclusione*: una risorsa può essere acquisita da un processo alla volta
  - *allocazione parziale (hold-and-wait)*: un processo richiede le risorse in più fasi
  - *assenza di pre-emption*: non è possibile sottrarre d'autorità una risorsa ad un processo
  - *attesa circolare*: esiste una catena chiusa di processi ciascuno in attesa di una risorsa posseduta dal prossimo processo della catena



Soluzione(1)

- Le risorse disponibili sono:

R1	R2	R3	R4
0	3	1	2

- Le risorse necessarie sono:

R1	R2	R3	R4
1	9	2	7

- Si può verificare un deadlock perché le risorse disponibili non sono sufficienti a soddisfare tutte le richieste dei processi contemporaneamente. Ad esempio la sequenza:
  P1 alloca 2 R4; P2 alloca 3 R2; P4 alloca 1 R3
  porta il sistema in una stato di deadlock.

risorse disponibili

R1	R2	R3	R4
0	0	0	0

allocazione

R1	R2	R3	R4
1	0	2	4
1	5	0	1
2	1	2	0
0	0	1	2

richieste

R1	R2	R3	R4
0	1	0	0
0	1	1	0
0	2	0	5
1	2	0	0

Algoritmo del banchiere

- La soluzione consiste nell'allocare le risorse secondo l'algoritmo del banchiere:
- Una risorsa richiesta viene assegnata ad un processo se l'assegnazione porta il sistema in uno stato sicuro, negata se porta il sistema in uno stato insicuro
  - uno stato è sicuro se esiste una sequenza di esecuzione di processi $P_i \ldots P_n$ in cui ogni processo può completare l'esecuzione
  - uno stato è insicuro se non esiste nessuna sequenza di esecuzione di processi che li completa tutti.

Soluzione(2)

- Cerchiamo una sequenza di esecuzione che porti ogni processo al suo completamento:

risorse disponibili

R1	R2	R3	R4
0	3	1	2

allocazione

R1	R2	R3	R4
1	0	2	2
1	2	0	1
2	1	2	0
0	0	0	2

richieste

R1	R2	R3	R4
0	1	0	2
0	4	1	0
0	2	0	5
1	2	1	0

Soluzione(3)

- Cerchiamo una sequenza di esecuzione che porti ogni processo al suo completamento:

risorse disponibili

R1	R2	R3	R4
0	2	1	0

allocazione

R1	R2	R3	R4
1	1	2	4
1	2	0	1
2	1	2	0
0	0	0	2

richieste

R1	R2	R3	R4
0	0	0	0
0	4	1	0
0	2	0	5
1	2	1	0

- Il processo P1 alloca 1 R2 e 2 R4 e termina la sua computazione.

risorse disponibili

R1	R2	R3	R4
1	3	3	4

Soluzione(4)

- Cerchiamo una sequenza di esecuzione che porti ogni processo al suo completamento:

risorse disponibili

R1	R2	R3	R4
0	1	2	4

allocazione

R1	R2	R3	R4
0	0	0	0
1	2	0	1
2	1	2	0
1	2	1	2

richieste

R1	R2	R3	R4
0	0	0	0
0	4	1	0
0	2	0	5
0	0	0	0

- Il processo P4 alloca 1 R1, 2 R2 e 1 R3 e termina la sua computazione.

risorse disponibili

R1	R2	R3	R4
1	3	3	6

Soluzione(5)

- Cerchiamo una sequenza di esecuzione che porti ogni processo al suo completamento:

risorse disponibili

R1	R2	R3	R4
1	1	3	1

allocazione

R1	R2	R3	R4
0	0	0	0
1	2	0	1
2	3	2	5
0	0	0	0

richieste

R1	R2	R3	R4
0	0	0	0
0	4	1	0
0	0	0	0
0	0	0	0

- Il processo P3 alloca 2 R2 e 5 R4 e termina la sua computazione.

risorse disponibili

R1	R2	R3	R4
3	4	5	6

Soluzione(6)

- Cerchiamo una sequenza di esecuzione che porti ogni processo al suo completamento:

risorse disponibili

R1	R2	R3	R4
3	0	4	6

allocazione

R1	R2	R3	R4
0	0	0	0
1	6	1	1
0	0	0	0
0	0	0	0

richieste

R1	R2	R3	R4
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

- Il processo P2 alloca 4 R2 e 1 R3 e termina la sua computazione.

risorse disponibili

R1	R2	R3	R4
4	6	5	7

Esercizio 2 – Gestione Risorse

- Quattro Processi, P1-P4, richiedono 4 risorse R1-R4. Si supponga che le risorse vengano allocate ad un ad una e che nessuna delle risorse venga rilasciata da un processo prima della terminazione. In un certo istante la fotografia del sistema è rappresentata dalle tabelle:

risorse esistenti

R1	R2	R3	R4
3	9	8	9

allocazione

R1	R2	R3	R4
1	0	0	0
1	2	3	4
0	2	2	2
0	0	1	3

richieste

R1	R2	R3	R4
0	7	5	0
1	0	0	2
0	0	2	0
0	6	4	2

Esercizio 2 a

- Il sistema è in uno stato sicuro?
  - Sì, perché la sequenza:

P3 alloca 2 R3; P4 alloca 6 R2, 4 R3 e 2 R4;
P1 alloca 7 R2 e 5 R3; P2 alloca 1 R1 e 2 R4
porta alla terminazione di tutti i processi.

risorse disponib.

R1	R2	R3	R4
1	5	2	0

allocazione

R1	R2	R3	R4
1	0	0	0
1	2	3	4
0	2	2	2
0	0	1	3

richieste

R1	R2	R3	R4
0	7	5	0
1	0	0	2
0	0	2	0
0	6	4	2

Esercizio 2 b

- Secondo l'algoritmo del banchiere, la richiesta di P4 di 5 R2 e 2 R3 può essere soddisfatta?
  - No, perché le risorse restanti non permettono di far concludere nessun processo.

risorse disponib.

R1	R2	R3	R4
1	5	2	0
1	0	0	0

allocazione

R1	R2	R3	R4
1	0	0	0
1	2	3	4
0	2	2	2
0	0	1	3

richieste

R1	R2	R3	R4
0	7	5	0
1	0	0	2
0	0	2	0
0	6	4	2

Esercizio 2 c

- Il processo P3 chiede 2 R3. Come evolve il sistema?
  - La richiesta può essere accettata perché il sistema rimane in uno stato sicuro dopo aver soddisfatto la richiesta. Il processo P3 termina.

risorse disponib.

R1	R2	R3	R4
1	5	2	0
1	7	4	2

allocazione

R1	R2	R3	R4
1	0	0	0
1	2	3	4
0	2	2	2
0	0	1	3

richieste

R1	R2	R3	R4
0	7	5	0
1	0	0	2
0	0	2	0
0	6	4	2

Esercizio 2 d

- Secondo l'algoritmo del banchiere, la richiesta di P1 di 3 R2 e 2 R3 può essere soddisfatta?
  - Sì, perché il sistema rimane in uno stato sicuro dopo aver soddisfatto la richiesta.

risorse disponib.

R1	R2	R3	R4
1	7	4	2
1	4	2	2

allocazione

R1	R2	R3	R4
1	0	0	0
1	2	3	4
0	0	0	0
0	0	1	3

richieste

R1	R2	R3	R4
0	7	5	0
1	0	0	2
0	0	0	0
0	6	4	2

Esercizio 2 e

- Successivamente P4 richiede 2 R3 e 2 R4: la richiesta può essere soddisfatta?
  - No, perché il sistema rimane in uno stato non sicuro dopo aver soddisfatto la richiesta.

risorse disponib.

R1	R2	R3	R4
1	4	2	2
1	4	0	0

allocazione

R1	R2	R3	R4
1	3	2	0
1	2	3	4
0	0	0	0
0	0	1	3

richieste

R1	R2	R3	R4
0	4	3	0
1	0	0	2
0	0	0	0
0	6	4	2

Esercizio 3 – Gestione della memoria

- Un programma in formato eseguibile è diviso in un codice di 200 KB, un'area dati di 126 KB e uno stack di 52 KB. Si ipotizzi che la macchina fisica possa indirizzare un massimo di 64 MB di memoria, che l'indirizzamento logico sia a 20 bit e che il programma sia tutto in memoria durante l'esecuzione. La memoria è organizzata a pagine di 16 KB.

  Quanto è lunga la tabella delle pagine per il programma?

  Quanti bit ha ogni suo elemento?

  Qual è il numero di pagine logiche del più lungo programma eseguibile?

Esercizio 3 – Soluzione errata

(200 KB (Codice) + 126 KB (Dati) + 52 KB (Stack)

= 378 KB

378 KB / 16 KB (Dimensione pagina) = 24 pagine

→ codice, dati e stack allocano le pagine autonomamente

Esercizio 3 – Soluzione

- Codice: 200 KB / 16 KB = 12.5 → 13 pagine
- Dati 126 KB / 16 KB = 7.88 → 8 pagine
- Stack: 52 KB / 16 KB = 3.25 → 4 pagine

- 64 MB / 16 KB = 4096 pagine contenute in memoria che richiedono 12 bit per essere indirizzate (2^{12}=4096).

- La tabella delle pagine contiene 25 elementi di 12 bit ciascuno.

Esercizio 3 – Soluzione (cont.)

- Il più lungo programma eseguibile occupa tutto l'indirizzamento logico:

  2^{20} B / 16 KB = 1 MB /16 KB = 64 pagine

  ed occupa quindi 64 pagine.

Esercizio 3 – Soluzione alternativa (bit)

- 64 MB di memoria richiedono 26 bit di indirizzamento (2^{26} = 64 MB).
- 16 KB di memoria richiedono 14 bit di indirizzamento (*offset*) (2^{14} = 16 KB).
- 26 bit – 14 bit = 12 bit → ogni elemento della tabella delle pagine contiene 12 bit.
- 20 bit – 14 bit = 6 bit → il programma più lungo occupa 2^6 = 64 pagine.

E se utilizziamo i segmenti?

- Supponiamo di risolvere l'esercizio precedente con memoria organizzata a segmenti lunghi al massimo 64 KB. Quanti registri di segmentazione sono necessari al programma durante l'esecuzione? Quanti bit deve avere ogni registro? Qual è il numero di registri di segmentazione necessari per il più lungo programma eseguibile?

Esercizio 3 – Soluzione segmenti

- Codice: 200 KB / 64 KB = 3.13 → 4 segmenti
- Dati 126 KB / 64 KB = 1.98 → 2 segmenti
- Stack: 52 KB / 64 KB = 0.81 → 1 segmenti

- 64 MB di memoria richiedono 26 bit di indirizzamento (2^{26} = 64 MB).
- 64 KB di memoria richiedono 16 bit di indirizzamento (2^{16} = 64 KB).
- Sono necessari 7 registri di 42 bit ciascuno.

Esercizio 3 – Soluzione (cont.)

- Il più lungo programma eseguibile occupa tutto l'indirizzamento logico:
- 2^{20} B / 64 KB = 1 MB /64 KB = 16 segmenti.
  (Oppure 20 bit – 16 bit = 4 bit → 2^4 = 16 segmenti).

- Il più lungo programma eseguibile necessita di 16 registri di segmentazione.

Esercizio 4 – Gestione della memoria

- Un calcolatore ha uno spazio di indirizzamento logico per ogni processo di 128 KB, divisi in pagine di 16 KB. Se un programma ha il codice di 35250 B, un'area dati di 68000 B e uno stack di 17000 B, può essere eseguito in memoria senza ricorrere a paginazione su disco? Qual è la dimensione massima delle pagine che consente al programma di essere eseguito rimanendo completamente in memoria?

Esercizio 4 – Soluzione(1)

- Dimensione pagina: 16 KB = 16384 B
- Numero di pagine residenti in memoria:
  128 KB / 16 KB = 8 pagine
- Codice: 35250 B / 16384 B = 2.15 → 3 pagine
- Dati: 68000 B / 16384 B = 4.15 → 5 pagine
- Stack: 17000 B / 16384 B = 1.04 → 2 pagine

- Il programma necessita di 10 pagine ma la memoria ne può contenere solo 8.

Esercizio 4 – Soluzione(2)

- Dimensione pagina: 8 KB = 8192 B
- Numero di pagine residenti in memoria:
  128 KB / 8 KB = 16 pagine
- Codice: 35250 B / 8192 B = 4.30 → 5 pagine
- Dati: 68000 B / 8192 B = 8.30 → 9 pagine
- Stack: 17000 B / 8192 B = 2.08 → 3 pagine

- Il programma necessita di 17 pagine ma la memoria ne può contenere solo 16.

Esercizio 4 – Soluzione(3)

- Dimensione pagina: 4 KB = 4096 B
- Numero di pagine residenti in memoria:
  128 KB / 4 KB = 32 pagine
- Codice: 35250 B / 4096 B = 8.61 → 9 pagine
- Dati: 68000 B / 4096 B = 16.60 → 17 pagine
- Stack: 17000 B / 4096 B = 4.15 → 5 pagine

- Il programma necessita di 31 pagine e la memoria ne può contenere 32, quindi il programma può essere eseguito rimanendo completamente in memoria.

Esercizio 5 – Gestione Risorse

- Un sistema ha 6 unità nastro con *n* processi che competono per ottenerle. Ogni processo può chiederne al più 2. Per quali valori di *n* il sistema è esente da deadlock nei seguenti casi:
  - i processi chiedono al più 2 unità qualunque
  - i processi chiedono 2 unità specifiche

  Si discutano le modalità di allocazione per evitare l'eventuale deadlock.

Esercizio 5 – Soluzione(1)

- Le condizioni di
  - mutua esclusione
  - allocazione parziale
  - assenza di preemption

  sono presenti, l'esercizio quindi mira ad evitare la condizione di attesa circolare.

Esercizio 5 – Soluzione(2)

- Primo caso: il sistema è esente da deadlock per

  $0 < n < 6$

- Secondo caso: il sistema è esente da deadlock per

  $n = 1$

- Strategie di allocazione per evitare il deadlock:
  - Algoritmo del banchiere
  - I processi possono richiedere le risorse solo in modo ordinato