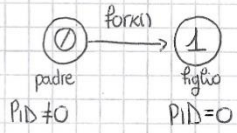
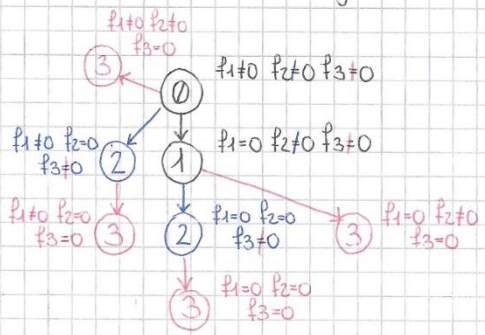
ESERCIZI-CREAZIONE DI PROCESSI



- pid = fork() pid è associato al padre 0 e figlio
- fork() || fork()
- fork() && fork()

- if(!fork()){ same as if(fork()==0)
  ↓ ma fork va eseguito e poi vedo la condizione
  // child
  }else{
  // parent
  }

- pid_t f1, f2, f3;
  - f1 = fork();
  - f2 = fork();
  - f3 = fork();



ESERCIZI - SCHEDULING DELLA CPU

Algoritmi:
- **FCFS** (First come, first served) => Possibile effetto convoglio
- **SJF (non preemtive)** si va dal processo a burst minore fino al maggiore
- **SRFT (preemtive)** » e si guarda il loro arrivo nel tempo, se è più breve si interrompe quello che era in esecuzione (più lungo)
- **RR** si dà un quanto q, superato il processo viene interrotto e si esegue il successivo (Tipo FCFS)

Ta: tempo di attesa — CPU / IO

Tempo risposta: Prima volta che eseguo la CPU

Tt: tempo di turnaround : tempo di esecuzione di un processo

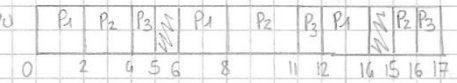
I/O:
- Un processo può tornare in CPU solo se ha terminato I/O e viceversa
- La coda di I/O è solitamente FCFS

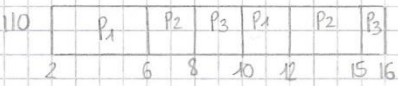
↓ esempio

Processo	B1	I/O 1	B2	I/O 2	B3	Arrivo
P_1	2	4	2	2	2	0
P_2	2	2	3	3	1	1
P_3	1	2	1	1	1	1

NB Nel tempo di attesa non va considerato quello della coda di I/O

FCFS

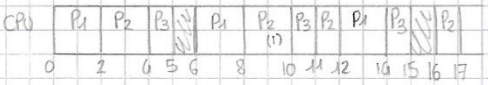


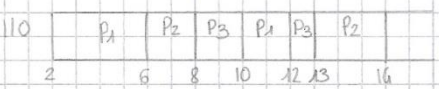


Ta: $P_1 = 0$, $P_2 = 1$, $P_3 = 4$ — $\overline{Ta} = \frac{0+1+4}{3} = 1{,}\overline{6}$ msec

Tt: $P_1 = 14$, $P_2 = 15 = 16-1$, $P_3 = 16 = 17-1$ — $\overline{Tt} = 15$ msec

RR q=2





Ta: $P_1 = 0$, $P_2 = 1+1 = 2$, $P_3 = 3+1 = 4$ — $\overline{Ta_m} = \frac{2+4}{3} = 2$ msec

Tt: $P_1 = 14$, $P_2 = 17-1 = 16$, $P_3 = 15-1 = 14$ — $\overline{Tt_m} = \frac{14+16+14}{3} = 14{,}\overline{6}$ msec

ESERCIZI - SINCRONIZZAZIONE

- PROBLEMA TIPO 1: Diverse codici di processi con una variabile condivisa
  - Indicare l'output (tutte le combinazioni)
  - Dove inserire wait()/signal()
  - Combinazione di valori dei semafori
  - Se ho race condition

Semafori

```
wait(S){                    signal(S){
  while(S<=0)                 S++;
    busy waiting            }
  S--;
}
```

- PROBLEMA TIPO 2: Produttore-consumatore, ho sempre un codice del tipo:

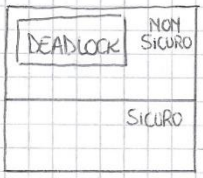
```
semaphore mutex=1, pieno=0, vuoto=0
int porzioni = M;

CUOCO while(true){
        wait(vuoto),
          <riempie la pentola>
        signal(pieno)
      }

CANNIBALE while(true){
            wait(mutex);
            if(porzioni==0){
              signal(vuoto);
              wait(pieno);
            }
            porzioni--;
            signal(mutex)
          }
```

ESERCIZI - DEADLOCK

- PROBLEMA TIPO 1: Il sistema è in stato sicuro? Quale è una sequenza sicura?
  - Need = Max - Allocation
  - Work ← Available
  - Needi ≤ work ? → NO i++
    - SI → work ← work + Allocation



- Se non ho available: totale istanze − totale di Allocation = Available
- Una ulteriore richiesta da parte di Pi: (1 0 1) (es) può essere accolta?

  Sommo ad Allocationi la richiesta
  sottraggo a Needi "
  " a Available "

  e ripeto l'algoritmo

- Posso avere delle variabili all'interno di Allocation e Max, x,y,z, devo trovare il range ammesso
  1. da Need, prima di partire, prendo le risorse con le variabili e le pongo ≥0
  2. Mentre svolgo l'algoritmo, Needi ≤ Available
     2-X ≤ 3 (es)

     2-X ≥0 (es)

     unendo ottengo il vincolo

  NB Nel mentre contratto sempre le variabili al caso peggiore, perché in caso non vada bene posso restringere l'intervallo per risolvere

- Se ho CARICO ATTUALE e RICHIESTE ATTUALI (=Allocation)
  1. Nuova allocation = CARICO ATTUALE + RICHIESTE ATTUALI
  2. Nuova Need = NEED - RICHIESTE ATTUALI

ESERCIZI – MEMORIA CENTRALE

- Allocazione di processi in memoria

FIRST FIT: è allocato il primo spazio abbastanza grande

BEST FIT: è allocato il buco più piccolo capace di contenere il processo

- Nodi: spazi liberi
- Frammentazione esterna: sommo gli spazi rimasti vuoti

- Paginazione

Spazio logico 2^n [B] da DIM PAG 2^m [B]

Memoria fisica 2^p [B]

* NB se unità sono la pagina / frame

⇒ | n | m |

| p | m |

	#pagina	offset	
LOGICO	n-m	m	[bit]
FISICO	p-m	m	[bit]

- Dimensione massima di un processo? $\text{DIM}_{\text{PROCESSO}}^{\text{MAX}} = 2^n \cdot 2^m$

- Data la tabella delle pagine, traduzo gli indirizzi logici in indirizzi fisici:

X → lo traduco in binario, lo divido in due pezzi: #pagina logica | offset

lo cerco sulla tabella e vedo il corrispettivo associato Y

traduco Y in binario | offset ⇒ indirizzo fisico

* - Dimensione della tab delle pag. invertita, se il sistema non supporta più di 2^x processi alla volta

PID + #pag. logica

x n (bit totali)

$\text{DIM}_{\text{INV.}}^{\text{TAB}} = 2^p \cdot (x+n)$ in byte!

2^x non è una potenza del 2 ⇒ approssima a quella che lo contiene

* - Calcolare i #pagina logica e offset dei seguenti indirizzi: X

X / R	2^n / Q	Q (binario) su n bit	R (binario) su m bit

Programma n [B] testo, m [B] dati, p [B] stack — pagine da Z [B] = $\text{DIM}_{\text{PAG.}}$ — Memoria da K [B] — spazio di indirizzamento a X bit

- Dimensione della tabella delle pagine? (quante è lunga? in pagine)

$\left[n/Z + m/Z + p/Z \right] \cdot X \text{ (in byte)} = \text{DIM}_{\text{TAB}}$ (arrotondo per eccesso)

- Quanti bit ha ogni suo elemento?

K/Z = # pag contenute in memoria che richiedono a bit per essere indirizzate $= 2^a$

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

$\frac{2^x}{Z}$ = # pagine

⇓

la tabella delle pagine ha DIM_{TAB} elementi di a bit ciascuno

ESERCIZI - MEMORIA VIRTUALE

- Sostituzione delle pagine

FIFO: sostituisco la pagina che si trova in testa alla coda

OTTIMO: sostituisco la pagina che non verrà usata per il periodo di tempo più lungo (futuro)

LRU: sostituisco la pagina che non è stata utilizzata da più tempo

CLOCK: aggiungo un bit 0/1
- Quando la pag. riceve una seconda chance, il bit è azzerato ed il tempo di arrivo viene aggiornato al tempo reale
- 0 → rimpiazzo e metto a 1
- 1 → lo pongo a 0, lascio la pagina, rimpiazzo la successiva (quella che ha 0)

ES

7 10 15 19 23 10 7 5 19 15 16 17 19 15 16 18 17 16 23 — 5 blocchi

LRU

7	7	7	7	17	17	17	17	17	17
10	10	10	16	16	16	18	18	18	18
15	5	5	5	5	16	16	16	16	16
19	19	19	19	19	19	19	19	16	16
23	23	15	15	15	15	15	15	15	23
6	1	1	1	1	1	1	1	1	

13 page fault

CLOCK

1 7	1 5	1 5	1 5	1 16	1 16	1 16	1 16
1 10	0 10	1 16	1 16	0 16	1 18	1 18	1 18
1 15	0 15	1 15	1 15	0 15	0 15	1 16	1 16
1 19	0 19	1 19	1 19	0 19	0 19	0 19	1 23
1 23	0 23	0 23	1 17	0 17	0 17	1 17	1 17

12 page fault

- Calcola la stringa dei riferimenti (*) ed il #frame

DIM PAG = Dp (parole)

DIM MEM = DM (parole)

$$* = \frac{\text{rif. memoria}}{Dp} \quad \text{(approssima per difetto)}$$

$$\#frame = \frac{DM}{Dp}$$

- Page fault

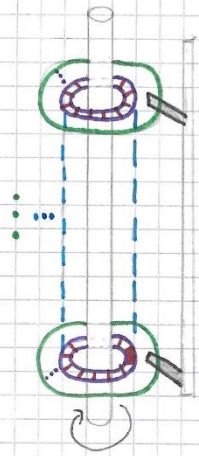
Page fault Rate: $0 \le p \le 1$ (↓ non ho PF, → ogni rif. è PF)

$$\text{Tempo medio di accesso effettivo } EAT = (1-p) \times t_{\text{accesso memoria}} + p\left[t_{\text{servizio PF}} + t_{\text{swap out}} + t_{\text{swap in}} + t_{\text{riavvio processo}}\right]$$

ES $t_{\text{accesso memoria}} = 200$ nsec $\quad t_{\text{servizio PF}} = 8$ msec $= 8000000$ nsec $\quad \Rightarrow EAT = 200 + 7999800p$ (nsec)

se 1 accesso su 1000 causa PF ($p = 1/1000$) $\Rightarrow EAT = 8{,}2$ μsec

ESERCIZI - MEMORIA SECONDARIA



insieme di tutti i sistemi di calcolo

C: cilindri t: tracce (anelli conc) S: settori testine

#piatti (dischi)

- #testine = #piatti
- Tot [B] per settore
- # settori per traccia

- **Tempi**

Tempo di accesso medio (posizionamento) = Seek time medio (distanza tra C(i) (ricerca): sposto il braccio del disco al cilindro desiderato) + Latenza media (di rotazione): il settore si posiziona sotto la testina

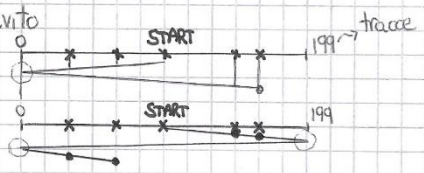
$\frac{60}{RPM} \cdot 500\ [m\ sec]$ → velocità di rotazione

$$\text{Tempo medio di I/O} = \text{Tempo medio di accesso} + \frac{\text{quantità dati da trasferire}}{\text{velocità di trasferimento}} + \text{overhead}$$

$\frac{\text{quantità dati da trasferire}}{\text{velocità di trasferimento}}$ → Tempo di trasferimento (x blocco)

- **Scheduling del disco**

- **FCFS**: il primo che arriva, è il primo ad essere servito
- **SCAN**: si servono sequenzialmente le richieste fino agli estremi
- **C-SCAN**: " ma arrivato alla fine, torna all'inizio senza servire eventuali richieste



- **LOOK**: ↘
- **C-LOOK**: ↗ analoghi, ma si fermano fin dove servono

$$\text{Tempo di servizio} = \sum_{i=1}^{N} \left[d \cdot \text{seek time} + \#\text{settori} \cdot (\text{t. latenza} + \text{t. trasf.}) \right]$$

(N: spostamenti, d)

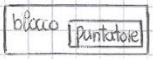
ESERCIZI - REALIZZAZIONE FILE SYSTEM

- Metodi di allocazione

- **CONTIGUA**: ciascun file occupa un insieme di blocchi contigui sul disco

- **CONCATENATA**: ciascun file è una lista concatenata di blocchi
  ↳ sparsi ovunque

  + FAT



- **INDICIZZATA**: per ogni file, si collezionano tutti i puntatori in un unico blocco indice
  - schema concatenato
  - indice a più livelli

  + UNIX

Tabella indice

12.30

- FAT

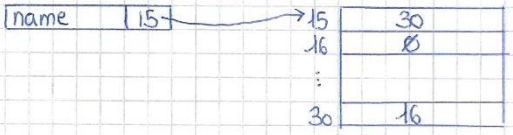
- Dimensione memoria secondaria D_M [B]
- Dimensione blocco D_B [B]
- Indirizzo blocco I_B [B]
- Numero di righe N_{FAT} FAT
- Dimensione D_{FAT} [B] della FAT
- Blocchi che occupa N_{BFAT} la FAT

$$N_{FAT} = \frac{D_M}{D_B} \qquad N_{FAT} = 2^{I_B\,(\text{in bit})} \qquad D_{FAT} = N_{FAT} \cdot I_B \qquad N_{BFAT} = \frac{D_{FAT}}{D_B}$$

- Quanti accessi per recuperare l'indirizzo fisico del blocco dove si trova il byte N?

$$\text{Blocco a cui appartiene il byte} = \frac{N}{D_B} = (\#) \rightarrow \text{, accessi}$$

- Blocchi fisici 15, 30, 16



• Accessi per cancellare/aggiungere

n record, posso crescere solo oltre l'n-esimo blocco

* $\frac{\text{DIM Blocco}}{\text{occ. Puntatore}}$ = # indirizzi che contiene il blocco

SEQUENZIALE

	CANCELLARE	AGGIUNGERE	LEGGERE
inizio	nessun accesso	n letture + n+1 scritture	
metà $n/2$	$n/2^{-1}$ letture + $n/2^{-1}$ scritture	$n/2$ letture + $\frac{n}{2}$+1 scritture	1 accesso
fine	nessun accesso	1 scrittura	

CONCATENATA

	CANCELLARE	AGGIUNGERE	
inizio	1 lettura	1 lettura + 1 scrittura	
metà $n/2$	$n/2^{+1}$ letture + 1 scrittura	$n/2$ letture + 1 scrittura	$n/2$ accessi
fine	n-1 letture + 1 scrittura	n letture + 1 scrittura	

INDICIZZATA

	CANCELLARE	AGGIUNGERE	
inizio	=	n letture + n+1 scritture	
metà $n/2$	1 lettura + 1 scrittura	$n/2$ letture + $n/2$+1 scritture	2 accessi * accesso unità blocco
fine	=	1 scrittura	

• Dimensione massima del file

- Dimensione blocco D_B [B]
- Indirizzi di blocco I_B [bit]

+ CONCATENATA: $\text{DIM}_{\text{FILE}}^{\text{MAX}} = 2^{I_B}(D_B - I_B[B])$

+ INDICIZZATA (a 2 livelli): $\#\text{puntatori} = \frac{D_B}{I_B[B]}$

$\text{DIM}_{\text{FILE}}^{\text{MAX}} = \#\text{punt} \cdot \#\text{punt} \cdot D_B$

+ # blocchi assegnati con file da N [B]

+ CONCATENATA: $D_{B_{\text{reale}}} = D_B - I_B[B]$ $\quad \#\text{blocchi} = \frac{N}{D_{B\,\text{reale}}}$

+ INDICIZZATA: $\#\text{blocchi} = \frac{N}{D_B}$

• Memoria sprecata

SEQUENZIALE: 0 sprechi

CONCATENATA: $\frac{\text{DIM FILE}}{\text{DIM Blocco}} \cdot \text{DIM puntatore}$ (x2 se doppiamente concatenato)

INDICIZZATA: DIM Blocco (• # in caso servano più tabelle (es. 2))

• UNIX (i-node)

n puntatori a blocchi diretti
m puntatori a singola indirezione
p puntatori a doppia indirezione

• Dimensione blocco D_B [B]
• Dimensione puntatore D_p [B] *

+ Dimensione massima file senza accessi aggiuntivi $DIM_{MAX} = n \cdot D_B$

+ Dimensione massima file $DIM_{MAX} = n \cdot D_B + m \cdot 2^{D_p} \cdot D_B + p \cdot 2^{D_p} \cdot 2^{D_p} \cdot D_B$

* se invece ho: indirizzi da TOT [B]

$$\text{puntatori per blocco} = \frac{D_B}{TOT} = A \qquad DIM_{MAX\ FILE} = n D_B + m \cdot A \cdot D_B + p \cdot A \cdot A \cdot D_B$$

$$DIM_{DISCO} = 2^{l_B(\text{in bit})} \cdot l_B\ [B]$$

+ Quanti blocchi sono utilizzati per allocare un file composto da N record, da M byte ciascuno? (i record non possono essere divisi in due blocchi)

$$\#\text{puntatori per blocco } A = \frac{D_B}{TOT} \qquad \#\text{record per blocco } B = \frac{D_B}{M} \text{ (approx. x difetto)} \Rightarrow \#\text{blocchi occupati dal file } C = \frac{N}{B}$$

+ Quanti blocchi sono necessari a gestire la sua allocazione?

- n puntatori dal nodo principale → rimangono $C - n = C'$
- A con singola indirezione → rimangono $C' - A = C''$
- doppia indirezione → $\frac{C''}{A} = \#$ (approx x eccesso)

1 blocco + 1 blocco + # → blocchi per gestire l'allocazione del file = D+1

+ Occupazione totale di memoria secondaria
↳ $(C+D)D_B$

DEADLOCK - TEORIA

- Il sistema è in stato sicuro? Quale è una sequenza sicura?
  - Need = Max - Allocation
  - work ← Available

se non ho available: totale risorse − totale allocation = available

  - Needi ≤ work? No ⇝ ciclo
  - Sì ⇝ work ← work + allocationi
  - ricomincio

21/05/2017 tirerei en G

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	2+x	1	4	0	4	3	8	2	2	1	2	1
P_1	0	6	2+y	4	4	8	6	6				
× P_2	4	6	4	2+z	5	7	8	4				
× P_3	x	3	3	3	2	5	6	5				
× P_4	1	2	1	2	3	3	2	3				

a) Il sistema è in stato sicuro?

- Need = Max - Allocation

	Need				Vincoli	
	A	B	C	D		
P_0	2-x	2	4	2	2-x ≥ 0	x ≤ 2
✓ P_1	4	2	4-y	2	4-y ≥ 0	y ≤ 4
✓ P_2	1	1	4	2-z	2-z ≥ 0	z ≤ 2
✓ P_3	2-x	2	3	2		
✓ P_4	2	1	1	1		

- Inizio l'algoritmo

P_4 ✓ Available + Allocation 4 = 3 3 3 3

P_3 ✓ $2-x \le 3 \Rightarrow x \ge -1 \Rightarrow -1 \le x \le 2$

Available + Allocation 3 = 3+x 6 6 6

P_2 ✓ $2-z \le 6 \Rightarrow z \ge -4 \Rightarrow -4 \le z \le 2$

= 7+x 12 10 8+z

P_1 ✓ $4-y \le 10 \Rightarrow y \ge -6 \Rightarrow -6 \le y \le 4$

= 7+x 18 12+y 12+z

P_0 $2-x \le 7+x$? nel peggiore x = −1

$2+1 \le 7-1$

$3 \le 6$ ✓ posso fare tutto ⇒ il sistema è in stato sicuro

b) all'istante richiesta P_1 con (1, 2, 2, 0)?

Ora Available: (1, −1, 0, 1) non possibile

Allocation: (1 8 4+y 4) Need: (3 0 2−y 2)

esercizio inclusione

	MAX A	B	C	Allocation A	B	C	Risorse disponibili A	B	C
P_0	7	4	3	0	1	0	10	4	7
P_1	3	2	2	3	0	2 ✓			
P_2	9	0	2	3	0	2			
P_3	2	2	2	② 2	2	1 ✓			
P_4	4	3	3	0	0	2			

× Need: 7 3 3 — Max - Allocation
0 2 0 ✓
6 0 0
② 0 1 1 ✓
4 3 1

× Available: 2 2 0 — ΣAllocation - R.d.

- È in stato sicuro?

P_1 ✓ 5 2 2
↓
P_3 ✓ 7 3 3
Posso fare tutto poi ✓

- Ulteriore richiesta P_3: (0, 1, 0)

Available diventa (2 1 0) : Available-P_3 * no, non posso neanche pazzie

13 LUGLIO 2017 es 2

	Allocation A	B	C	D	Max A	B	C	D	Available A	B	C	D
P_0	2	1	0	2	12	7	12	8+x	1	1	2	1
× P_1	2	2	2	6	11	6	12	9				
× P_2	2	1	1	4	9	4	10	6				
× P_3	1	1	1+4	0	7	3	8	2				
× P_4	2+z	1	1	1	6	2	2	2				

- Intervalli di x,y,z per i quali il sistema è in stato sicuro

Need = Max - Allocation

	A	B	C	D		
✓ P_0	10	6	12	6+x	6+x ≥ 0	x ≥ -6
✓ P_1	9	4	10	3		
✓ P_2	7	3	9	2		
✓ P_3	6	2	7-y	2	7-y ≥ 0	y ≤ 7
✓ P_4	4-z	1	1	1	4-z ≥ 0	z ≤ 4

P_4: $4-z \leq 1 \rightarrow z \geq 3$ $\quad 3 \leq z \leq 4$

P_3 $7-y \leq 3 \rightarrow y \geq 4$ $\quad 4 \leq y \leq 7$ (cambio) $\rightarrow 5 \leq y \leq 7$

P_2:

P_1:

P_0 $6+x \leq 12 \rightarrow x \leq 6$ $\quad -6 \leq x \leq 6$

Available			
3+z	2	3	2
4+z	3	4+y	2
6+z	4	5+y	6
8+z	6	7+y	12
10+z	7	7+y	16

se y=4 non posso fare nulla

- ult. richiesta P_3 (0, 0, T, 0) Trova T in modo che lo stato sia sicuro?

$R \leq N$ $\quad T \leq 7-y$
$R \leq Av$ $\quad T \leq 2$

	Allocation (+)				Need (−)				Available (−)			
P_3	1	1	1+4+T	0	6	2	7-y-T	2	1	1	2-T	1

1 Giugno 2022 es1 II Prova in itinere

Esercizio 1 (8 punti) DEADLOCK

Supponendo che cinque processi, P1, P2, P3, P4 e P5, condividano un insieme di risorse di quattro tipi diversi, A, B, C, D, si ipotizzi di trovarsi, ad un certo istante t, nella seguente situazione:

Allocation MAX Avaible

Processo	Carico attuale	Richieste massime	Richieste attuali	Risorse disponibili
	A B C D	A B C D	A B C D	A B C D
P1	1 0 2 0 √	3 2 4 2	X 2 Y 2	3 4 0 1
P2	0 3 1 2 √	3 5 1 2		
P3	2 4 5 1	2 7 7 5		
P4	3 0 0 6	5 5 0 8	2 X 0 2	
P5	4 2 1 3 √	6 2 1 4	2 0 0 Z	

Determinare (se esistono) quali sono i valori di X, Y e Z tali che le richieste possano essere soddisfatte, presentando almeno una successione sicura che garantisca l'assenza di stallo; in caso contrario, mostrare come i cinque processi potrebbero entrare in deadlock.

Trovo Need = Max - Allocation

	A	B	C	D
P_1	2	2	2	2
P_2	3	2	0	0
P_3	0	3	2	6
P_4	2	5	0	2
P_5	2	0	0	1

Uso le richieste attuali

Nuova Allocation + Nuova Need

	Nuova Allocation A	B	C	D	Nuova Need A	B	C	D
P_1	1+X	2	2+Y	2 √	2-X	0	2-Y	0 √
P_2	0	3	1	2 √	3	2	0	0 √
P_3	2	6	5	1 √	0	3	2	6 √
P_4	5	X	0	8 √	0	5-X	0	0 √
P_5	6	2	1	3+Z √	0	0	0	1-Z √

$\begin{cases} 2-X \geq 0 \\ 2-Y \geq 0 \\ 5-X \geq 0 \\ 1-Z \geq 0 \end{cases} \Rightarrow \begin{matrix} X \leq 2 \\ Y \leq 2 \\ Z \leq 1 \end{matrix}$

Avaible

	A	B	C	D
P_2	3	7	1	3

P_5 $1-Z \leq 3$ $Z \geq -2$ $-2 \leq Z \leq 1$ √ — 9 9 2 6+Z

P_4 $5-X \leq 9$ $X \geq -4$ $-4 \leq X \leq 2$ √ — 16 9+X 2 16+Z

P_3 — 16 8+X 7 15+Z

P_1 $2-X \leq 16$ $X \geq -16$

$+2-Y \leq 7$ $Y \geq -5$ $-5 \leq Y \leq 2$ — 17+X 15+X 9+Y 17+Z

Esercizio 2 (6 punti)
Si supponga che cinque processi, A, B, C, D ed E, condividano un insieme di risorse di quattro tipi diversi, R_1, R_2, R_3, R_4.
Si supponga inoltre di trovarsi nella configurazione seguente:

		Allocation	Need	
√	A	0011	0020	√
	B	2100	0211	
√	C	0010	2100	√
√	D	2001	0110	√
	E	0200	3010	

e che le risorse totali siano rappresentate dal vettore R=[4,4,3,2]. Esiste una situazione di stallo? Se sì, quali sono i processi coinvolti? Quali e quante risorse addizionali permetterebbero di evitare lo stallo? Motivare la risposta data.

Available = [4,4,3,2] − [4,3,2,2] = [0,1,1,0]

	Need ≤ Available	Nuova available
D	√	2 1 1 1
C	√	2 1 2 1
A	√	2 1 3 2 → Non posso servire né B, né E ⇒ STALLO

Esercizio 4 (5 punti)
Si supponga che in un sistema multiprocessore a due processori siano presenti cinque processi, P_0 e P_1, allocati al processore 0, P_2, P_3 e P_4, allocati al processore 1, ed un insieme di risorse di quattro tipi diversi, A, B, C, D. Si supponga di trovarsi nella configurazione seguente:

	Allocation	Max	Available
P_0	0 x–1 0 2	2 4 5 6	2 2 10 4
P_1	10 0 y–2 2	12 7 6 8	
P_2	6 0 2 0	8 2 8 4	
P_3	2 0 3 2	2 3 4 2	
P_4	4 1 z+1 4	11 16 9	

Determinare:
- gli intervalli dei valori (interi) di x, y e z per i quali il sistema si trova in uno stato sicuro;
- se la richiesta ulteriore di P_2 pari a [2,0,1,1] può essere immediatamente soddisfatta, lasciando il sistema in stato sicuro.

a)

Need =	A	B	C	D
√ P_0	2	5-X	5	4
√ P_1	2	7	8-Y	6
√ P_2	2	2	6	4
√ P_3	0	3	1	0
√ P_4	7	0	5-Z	5

$\begin{cases} 5-X \geq 0 \\ 8-Y \geq 0 \\ 5-Z \geq 0 \end{cases} \rightarrow \begin{cases} X \leq 5 \\ Y \leq 8 \\ Z \leq 5 \end{cases}$

^	intervallo	N ≤ Av	Available + Allocation
P_2		√	2 2 10 4
P_0	$5-X \leq 2 \rightarrow X \geq 3$	√	8 2 12 4
P_3		√	8 X+1 12 6
P_4	$5-Z \leq 15 \rightarrow Z \geq -10$	√	10 X+1 15 8
P_1 v	$8-Y \leq Z+16 \rightarrow Y \geq 2$ (Z = −10)	√	14 X+2 Z+16 12
			24 X+2 Y+Z+16 16

$3 \leq X \leq 5 \Rightarrow X = 5$

$-10 \leq Z \leq 5$

$2 \leq Y \leq 8$

b)

	Allocation +				Need −				Available −			
P_2	8	0	3	1	0	2	5	3	0	2	9	3

	Available + Allocation			
P_2	8	2	12	4

Si posso eseguirlo

MEMORIA CENTRALE

1 GIUGNO 2022 es 3 II itinere

Esercizio 3 (6 punti)

Un programma in formato eseguibile è diviso in un codice di 230 KB, un'area dati di 118 KB e uno stack di 56 KB. Si ipotizzi che la macchina fisica indirizzi un massimo di 64 MB di memoria, che l'indirizzamento logico sia a 32 bit e che il programma sia tutto in memoria durante l'esecuzione. Se la memoria è organizzata a pagine di 4 KB:

1) Quanto è lunga la tabella delle pagine per il programma?
2) Quanti bit ha ogni suo elemento?
3) Qual è il numero di pagine logiche del più lungo programma eseguibile?

1) $\left[\frac{230K}{4K} + \frac{118K}{4K} + \frac{56K}{4K}\right] = 58+30+14 = 102$ pagine → La tabella delle pagine contiene 102 elementi di 14 bit ciascuno

2) $\frac{64MB}{4KB} = 16384$ pagine contenute in memoria che richiedono 14 bit per essere indirizzate ($2^{14} = 16384$)

3) $\frac{2^{32}B}{4KB} = 1M$ pagine

Esercizio 3 (8 punti)

Su una data architettura, ad ogni processo sono garantiti 65536 byte di spazio di memoria fisica, divisi in pagine di 4096 byte. Per il processo *P*, il segmento di testo consta di 32768 byte, i dati occupano 16386 byte e lo stack 15870 byte. Il processo *P* potrà essere eseguito sull'architettura descritta? Cosa accade se la dimensione della pagina fisica fosse invece pari a 512 byte? Si noti che nella stessa pagina non possono coesistere segmenti diversi.

ogni processo 65536B DIM PAGINA 4096B testo 32768B, dati 16386B, stack 15870B

a)

#BLOCCHI TOT: $\frac{65536B}{4096B} = 16$

#BLOCCHI NECESSARI: $\left[\frac{32768B}{4096B} + \frac{16386B}{4096B} + \frac{15870B}{4096B}\right] = 8+5+4 = 17$ → non ho abbastanza blocchi

b) #TOT $\frac{65536}{512} = 128$

$\#\text{necessari} = \left[\frac{32768B}{512B} + \frac{16386B}{512B} + \frac{15870B}{512B}\right] = 64+33+31 = 128$ → ho lo spazio giusto

---

Esercizio 3 (6 punti)

Un computer è dotato di uno spazio di indirizzi virtuali a 32 bit, con pagine da 8KB. La tabella delle pagine per ciascun processo non in esecuzione risiede sulla memoria di massa. Quando un processo inizia la propria esecuzione la tabella delle pagine viene caricata in RAM alla velocità di 4 byte ogni 100nsec. Se ciascun processo viene eseguito per 100msec, considerando anche il tempo necessario al caricamento della tabella delle pagine, quale frazione del tempo di CPU viene "sprecata" in questa operazione (sui 100msec assegnati al processo)?

DIM PAG $2^{13}B$

spazio logico $2^{32}B$

32 bit da 8kB 100msec

4B / 100nsec

Tempo per i processi $100m = 10^{-1}$ sec

Per spazio virtuale: | #pagina 19 | offset 13 | bit

DIM TAB PAGINE $2^{19} \cdot 4B = 2^{21}$

$\rightarrow \frac{2^{21}}{2^2} = 2^{19}$ $\Rightarrow 2^{19}(100 \cdot 10^{-9}) = 2^{19} \cdot 10^{-7} = 0{,}52 \cdot 10^{-6} \cdot 10^{-7} = 0{,}52 \cdot 10^{-1}$ sec

52% di spreco

Esercizio 4 (6 punti)

Si consideri un sistema con memoria virtuale e segmentazione con parola da un byte. Supponendo di avere indirizzi logici di 10 bit, i cui 4 più significativi indicano il numero di segmento, dire:

- qual è la massima grandezza possibile per un segmento;
- di quanti segmenti, al più, può essere composto un processo.

Supponendo che il processo attualmente in esecuzione sia composto da 5 segmenti, S_0, S_1, S_2, S_3, S_4, e che, ad un dato istante, la sua tabella dei segmenti sia la seguente:

Numero segmento	Lunghezza	Base
0	12	100
1	23	200
2	6	252
3	32	683
4	28	720

tradurre in indirizzi fisici i seguenti indirizzi logici: 8, 132, 79, 63, 259, 135, 321.

a) #Pagina 10-6 = 6 MAX SEGMENTO = $2^6 = 64$

b) #segmenti a processo $2^4 = 16$

c) 132: (10 bit) 0010 000100 → (2; 4) → 252 + 4 = 256;
(0010 = #segmento : 2; 000100 = 4)

79: 0001 001111 → (1; 15) → 200 + 15 = 215
(0001 = 1; 001111 = 15)

135: 0010 000111 → (2; 7) → 776 segmentation fault
(0010 = 2; 000111 = 7)

321: 0101 000001 → invalid segment
(0101 = 5)

Esercizio 4 (8 punti)

Un computer, dotato di un sistema per la gestione della memoria virtuale, utilizza pagine da 4KB. Lo spazio degli indirizzi virtuali è di 2^{32} byte, mentre la memoria fisica è costituita da 2^{18} byte. Si consideri la seguente tabella delle pagine relativa ad un particolare processo:

Frame	Present bit	Modified bit
1	1	0
7	0	0
9	1	1
14	1	0
8	1	0
3	1	1
25	0	0
16	0	1

1) Qual è il formato dell'indirizzo virtuale?
2) Qual è l'indirizzo fisico corrispondente all'indirizzo virtuale 0x00005B83?
3) Qual è la dimensione del processo?
4) Quanto spazio occupa la relativa tabella delle pagine (come sopra descritta)?

(su internet non ho trovato nulla)

DIM PAGINA = 4KB = 2^{12} B

SPAZIO LOGICO = 2^{32} B

MEMORIA = 2^{18} B

a) #pagina (20 bit) | offset (12 bit) ⇒ DIM PROCESSO $2^{20} \cdot 2^{12} = 2^{32}$ B

Esercizio 2 (10 punti)
Si consideri uno spazio degli indirizzi logici costituito da 128 pagine da 2KB, mappato su una memoria fisica di 64 frame. Quanti bit servono per descrivere lo spazio degli indirizzi logici? E quello degli indirizzi fisici? Qual è la dimensione massima di un processo? Qual è la dimensione della tabella delle pagine invertita supponendo che il sistema non supporti l'esecuzione contemporanea di più di 1000 processi alla volta? Calcolare numeri di pagina logica e offset relativi agli indirizzi seguenti (in decimale): 3315, 18363, 10000, 512, 16385.

Logico 128 = 2^7 pagine da 2KB = 2^{11} B
Fisico 64 frame = 2^6

1) spazio logico: #pagina 7 | offset 11 => 18 bit
spazio fisico: 6 | 11 => 17 bit

2) DIMMAX PROCESSO = 2^{18} = 256 KB

3) PID + #pagina logica = 10 + 7 = 17 bit = 2,125 B
1000 ≃ 2^{10} (1024)
=> DIMTAB INVERTITA = 2^6 (2,125 B) = 136 B

a) 3315 | 128
115 | 25
=> 25 su 7 bit | 115 su 11 bit
000 11 00 1 | 000 01 11 00 11

Esercizio 3 (5 punti)
Si consideri un sistema di memoria virtuale con le seguenti proprietà: indirizzi logici a 40 bit, pagine di 16KB, indirizzi fisici a 36 bit. Qual è la dimensione massima in MB della tabella delle pagine relativa ad un processo, supponendo che, come ulteriore informazione relativa a ciascuna pagina, vi siano un bit di validità, un bit di protezione, un bit di modifica ed un bit di uso (relativo ad un dato time slice)? Nelle stesse ipotesi, qual è il numero di elementi presenti nella tabella delle pagine invertita?

DIM RAM = 2^{36}

indirizzi logici 40 bit pagine da 16 KB = 2^{14} B
indirizzi fisici a 36 bit

a) indirizzo logico: #pagina 26 | offset 14 = 40
indirizzo fisico: #frame 22 | 14 = 36

DIMTAB (senza i bit di controllo) = 2^{26}

2^{26} · GB = 256 MB
22 bit + 1 + 1 + 1 + 1 = 26 bit ≃ GB

b) DIMTAB INVERTITA = 2^{22} · GB = 16 MB
26 bit + 1 + 1 + 1 + 1 + 1 = 30 bit ≃ GB

Esercizio 4 (6 punti)

Si consideri un sistema a memoria virtuale con indirizzi a 32 bit, indirizzi fisici a 20 bit e pagine da 512 byte, calcolare:

a) la struttura dell'indirizzo virtuale e dell'indirizzo fisico;
b) il numero delle pagine negli spazi di indirizzi virtuale e fisico;
c) quante pagine occupa un programma di 2100 byte?

Indirizzi logici 32 bit pagine da $512\,B = 2^9 B$
Indirizzi fisici 20 bit

a)

	#pagina	offset	
logico	23	9	32 bit
fisico	11	9	20 bit

Spazio virtuale $= 2^{32} B$ (ignoriamo la dimensione delle pagine 2^{23})
Spazio fisico $= 2^{20} B$

b) #pagine virtuali $= 2^{23}$ pagine
#pagine fisiche $= 2^{11}$ pagine

c) 2100 B

#pagine occupate dal processo $= \frac{2100B}{4KB} = 0{,}51 \sim 1$

NB parzialmente occupata
$S = 1 \cdot 4K - 2100 = 1996$
$\downarrow$
$\frac{1996}{4K} \simeq 48\%$ frammentazione interna

MEMORIA VIRTUALE

1 GIUGNO 2022 es 2 II prova in itinere

Esercizio 2 (5 punti) MEM. VIRT.

Si supponga di avere un sistema dotato di memoria virtuale con tempo medio di accesso alla memoria pari a 50nsec. Quando si verifica un page fault, il tempo di servizio è diverso nei due casi in cui esiste effettivamente un frame libero nel quale caricare la pagina richiesta o, viceversa, se deve essere selezionata una vittima, ed è pari, rispettivamente, a 25msec e 50msec. Supponiamo che, nella pratica, non esistano frame liberi nel 75% dei casi. Si calcoli il tasso di page fault massimo che assicuri comunque un tempo effettivo di accesso alla memoria inferiore a 250nsec.

$t_{accesso\ memoria} = 50\,nsec$ $t_{servizio} = t_{servizio\ (libero)} + t_{servizio\ (vittima)}$ $= 25m\,sec + 50m\,sec$ non ci sono frame liberi nel 75%

p? $EAT < 250\,nsec$

$$(1-p)\,50n + p\left[25m \cdot 0{,}25 + 50m \cdot 0{,}75\right] \leq 250n$$

$$50 \cdot 10^{-9} - 50 \cdot 10^{-9} p + 0{,}043\,p \leq 250 \cdot 10^{-9}$$

$$0{,}0429\,p \leq 2 \cdot 10^{-7} \rightsquigarrow p \leq 6{,}66 \cdot 10^{-6}$$

Esercizio 3 (7 punti)

Un computer ha quattro frame, i cui istanti di caricamento, di ultimo riferimento e i reference bit sono riportati nella seguente tabella:

Frame	Tempo di caricamento	Tempo di riferimento	Reference bit
2	135	287	1
1	240	250	1
0	169	253	0
→ 3	203	266	1

Dire quale pagina verrebbe liberata dall'algoritmo FIFO, da LRU e da CLOCK (in questo caso si supponga che l'ultimo frame controllato sia il 3).

FIFO coda: 2, 0, 3, 1 (in base al tempo di caricamento) → deve questo

LRU coda: 2, 3, 0, 1 (in base al tempo di riferimento) → deve questo

CLOCK coda: 2, 0, 3, 1 dà una seconda chance e alle pagine col bit a 1

→ 3 ① ⇒ ⓪
→ 1 ① ⇒ ⓪
→ 2 ① ⇒ ⓪
→ 0 deve questo

Esercizio 3 (6 punti)

Si consideri la seguente sequenza di riferimenti alla memoria di un processo, in un sistema con memoria virtuale:

2 1 3 4 2 1 5 7 2 6 1 7 3 6 1

Descrivere il comportamento degli algoritmi LRU e Clock di sostituzione delle pagine per una memoria fisica composta da 4 frame. Valutare, in entrambi i casi, il numero di page fault.

LRU

2°	2	2°	2	2	3
1°	1	1	6	6	6°
3	5	5	5	1	1°
6	6	7	7	7°	7
6	6	7	7	7	7

9 PF

CLOCK

1 2 ←	1 5	1 5	1 5	1 5 ←	1 1	1 1
1 1	0 1 ←	1 7	1 7	1 7	1 0 7 ←	1 7
1 3	0 3	0 3 ←	1 2	1 2	0 2 ←	1 3
1 6	0 6	0 6	0 6 ←	1 6	0 6	1 0 6 ←
6	1	1	1	1	1	1

10 PF

Esercizio 1 (8 punti)
Si consideri un sistema con 8 pagine di memoria fisica sul quale sono attivi tre processi A, B e C. Si supponga che all'istante 1000 siano caricate le pagine:

Processo	Pagina logica	Pagina fisica	Istante di caricamento	Riferimento
A	0	7	500	SI
A	8	6	480	NO
A	5	5	300	SI
B	5	4	970	NO
B	6	0	120	SI
C	2	2	765	NO
C	5	1	900	NO
C	9	3	212	NO

Supponendo che il sistema utilizzi un algoritmo di sostituzione seconda chance globale, dire cosa avviene nei seguenti casi:
1. C riferisce la pagina logica 5;
2. A riferisce la pagina logica 3.

Nota: Si noti che le pagine dei processi sono organizzate in una coda circolare FIFO.

La lista delle pagine per l'algoritmo second chance GLOBALE è

B6(120, Si)
C9(212, No)
A5(300 Si)
A8(480, NO)
A0(500, Si)
C2(765, NO)
C5(900, NO)
B5(970 NO)

1) la pagina C5 è già in memoria (nella pagina fisica 1) e quindi viene marcata come riferita
2) la pagina logica A3 non è in memoria, l'algoritmo second chance esamina B6, siccome è stata riferita, viene inserita in testa alla lista ed il campo riferita viene resettato, quindi viene esaminata C9, che viene eliminata, la lista risultante è

A5(300,Si) A8(480,NO) A0(500 Si) C2(765, NO) C5(900,NO)
B5(970,NO) B6(120 NO) A3(1000 Si)

Esercizio 3 (8 punti)
Si consideri la seguente sequenza di riferimenti in memoria nel caso di un programma di 480 parole:
8, 21, 107, 173, 63, 309, 184, 243, 248, 439, 458, 363

a) Si determini la stringa dei riferimenti supponendo che la dimensione delle pagine sia di 100 parole. Si calcoli inoltre il numero di page fault nel caso in cui la memoria principale a disposizione per il programma sia di 200 parole e si utilizzi LRU come algoritmo di sostituzione delle pagine
b) Si ripeta il procedimento supponendo che la dimensione delle pagine sia ridotta a 50 parole mentre la dimensione della memoria a disposizione e l'algoritmo di sostituzione rimangano invariati.
– Le dimensioni delle pagine utilizzate nell'esercizio sono irrealistiche; normalmente, infatti, la dimensione delle pagine è una potenza di due. Sapete spiegare perché?

a) 0 0 1 1 0 3 1 2 2 4 4 3 — 2 frame

0°	0	1	1	4°	4
1°	3	3	2°	2	3
2	1	1	1	1	1

7 PF

b) 0 0 2 3 1 6 3 4 4 8 9 7 — 4 frame

0°	6	6°	6	9	9
2	2	4	4	4	4
3	3°	3	3	3	7
1	1	1	8	8	8
4	1	1	1	1	1

9 PF

REALIZZAZIONE FILE SYSTEM

1 GIUGNO 2022 es 6 II finezze

Esercizio 4 (10 punti) FILE SYSTEM

Si consideri un file system gestito, secondo il modello UNIX, tramite i–node. Ciascun i–node contiene però 12 puntatori diretti a blocchi di dati e solo due puntatori a nodi indice, a singolo e a doppio livello di indirezione. Sapendo che la dimensione dei blocchi è 2KB e che gli indirizzi di blocco sono espressi con 64 bit, calcolare quanti blocchi verranno utilizzati per allocare un file composto da 11000 record, da 80 byte ciascuno, sapendo che i record non possono essere suddivisi su due blocchi. Si valuti inoltre il numero di blocchi necessari a gestire la sua allocazione e, conseguentemente, l'occupazione totale di memoria secondaria, relativa al file in oggetto, derivante dalla strategia di memorizzazione descritta.

12 diretti
1 singolo
1 doppio

$DB = 2KB = 2^{11} B$

$IB = 64 bit = 8B$

a) 11000 record da 80B

$\#\text{record per blocco} = \frac{2^{11} B}{80B} = 25$

$\#\text{blocchi occupati dal file} = \frac{11000}{25} = 440$

b) # blocchi necessari a gestire la sua allocazione?

- 12 nodo principale → 1 blocco
  440 − 12 = 428
- 256 a singola → 1 blocco
  428 − 256 = 172
- $172/256$ = 1 blocco

→ 3 blocchi + 1 = 4

$\#\text{Puntatori per blocco} = \frac{2^{11} B}{8B} = 256$

c) occupazione memoria?

(440 + 4) · 2^{11} B = 888 KB

1 GIUGNO 2022 es5 II finezze

Esercizio 5 (6 punti) FILE SYSTEM

Supponiamo che un file system supporti l'allocazione di file contigua, concatenata e indicizzata a singolo livello e supponiamo di avere appena letto l'informazione relativa al file nella directory (presente in RAM). Se si vuole leggere il contenuto del decimo blocco del file, quanti accessi alla memoria di massa sono necessari nei tre casi? Se si utilizza una bitmap per tenere memoria dello spazio libero su disco, con un disco da 40GB e blocchi da un 1KB, quale sarà la dimensione del vettore?

contigua 1 accesso
concatenata 10 accessi
indicizzata 2 accessi

DIM VETTORE $\frac{40 \cdot 2^{30}}{2^{10}} = 40M bit = 5MB$

Esercizio 4 (8 punti)
Si consideri una memoria di massa di 2^{36} byte, con blocchi da 2KB. Si utilizzi una tecnica di allocazione indicizzata per la memorizzazione dei file con 4 byte per gli indirizzi di blocco.

1. Quante operazioni di I/O occorrono per leggere il byte 104800 di un file da 400KB?
2. Se la maggior parte dei file occupa un solo blocco ed i rimanenti file occupano due blocchi, lo spazio "sprecato" in questo caso è maggiore o minore rispetto all'allocazione contigua e concatenata?
3. Nelle stesse ipotesi, il tempo di accesso a file è più o meno breve rispetto all'allocazione contigua e concatenata?

DIM MEMORIA = 2^{36} B allocazione indicizzata 4B indirizzi di blocco

DIM BLOCCHI = 2KB

1. $\#punt = \frac{2KB}{4B} = 512$ DIM MAX FILE = $512 \cdot 2KB = 1MB$

$\#blocchi\ assegnati\ al\ file = \frac{400KB}{2KB} = 200$

alessiadonati00@gmail.com

Esercizio 2 (7 punti)
Si consideri un disco da 64GB e si calcoli:
a) la dimensione della FAT16 (indirizzi composti da 16 bit) quando si impiegano blocchi di dimensioni minime (cioè si impiega il massimo numero di blocchi indirizzabili con questo tipo di FAT); si calcoli inoltre la dimensione del blocco;
b) La dimensione della FAT32 quando si impiegano blocchi di 2KB.

$D_H = 64GB$

a) $I_B = 16\,bit = 2B$ dimenzione FAT? dimenzione del blocco?

$D_{FAT} = 2^{16} \cdot 2 = 128\,KB \Rightarrow D_B = \frac{64GB}{2^{16}} = 1048576B = 1MB$

b) dimenzione FAT? $D_B = 2KB$

$N_{FAT} = \frac{64GB}{2KB} = 32M \quad D_{FAT} = 32M \cdot 4B = 128MB$

Esercizio 4 (8 punti)
Si consideri un file da 250KB allocato, utilizzando l'allocazione concatenata, in blocchi da 1KB, con puntatori lunghi 2 byte. Quanti accessi a disco sono necessari per cancellare il centesimo blocco del file? Quanto spazio viene "sprecato" per la memorizzazione dei puntatori in caso di lista concatenata semplice? E nel caso di una lista doppiamente concatenata? Viceversa, supponendo di utilizzare l'allocazione indicizzata, quanta frammentazione interna si ha nel blocco indice?

file 250 KB allocazione concatenata blocchi da 1KB Puntatori 2B

a) 100 letture + 1 scrittura

b) $\#blocchi = \frac{250KB}{1KB} = 250$ spazio sprecato $= 250 \cdot 2B = 500B$

c) spazio sprecato $= 250 \cdot 2B \cdot 2 = 1000B$

d) Blocco indice occupa 500B fr. interna $= 1KB - 500B = 524B$

Esercizio 5 (6 punti)
Calcolare la dimensione in byte della FAT per un disco da 4GB con blocchi da 16 KB e indirizzi dei blocchi a 32 bit.
a) Quanti blocchi occuperebbe la FAT se memorizzata su disco?
b) Quanti accessi alla FAT occorrono per recuperare l'indirizzo fisico del blocco nel quale si trova il byte 138831 di un certo file del file system in questione?

DIM DISCO = 4GB DIM BLOCCHI = 16KB Indirizzi a 32 bit = 4B

$N_{FAT} = \frac{4GB}{16KB} = 262144 \quad D_{FAT} = 262144 \cdot 4B = 1048576B = 1MB$

a) $N_{BFAT} = \frac{1MB}{16KB} = 64$

b) byte 138831 blocco a cui appartiene $= \frac{138831B}{16KB} = 8$ → 8 accessi

Esercizio 4 (6 punti)
Si consideri un file formato da 65 record e le sue possibili allocazioni su disco di tipo sequenziale, mediante lista concatenata e con tabella indice. Si supponga che l'FCB ed il blocco indice, nel caso di allocazione indicizzata, siano già stati caricati in RAM, così come eventuali dati da aggiungere al file. Si assuma inoltre che, in caso di allocazione sequenziale, il file possa "crescere" solo oltre il sessantacinquesimo blocco (i blocchi precedenti a quelli occupati dal file sono occupati). Si dica quanti accessi al disco sono necessari, in ognuna delle seguenti situazioni:
1) Cancellazione di un blocco dall'inizio, dalla metà e dalla fine del file;
2) Aggiunta di un blocco all'inizio, in mezzo ed alla fine del file.

65 record
metà → 32

SEQUENZIALE

	CANCELLARE	AGGIUNGERE
inizio	nessun accesso	65 letture + 66 scritture
metà	32 letture + 32 scritture	32 letture + 33 scritture
fine	nessun accesso	1 scrittura

CONCATENATA

inizio	1 lettura	1 lettura + 1 scrittura
metà	32 letture + 1 scrittura	32 letture + 1 scrittura
fine	66 letture + 1 scrittura	65 letture + 1 scrittura

INDICIZZATA

inizio	1 lettura + 1 scrittura	65 letture + 66 scritture
meta	↓	32 letture + 33 scritture
fine	↓	1 scrittura

Esercizio 3 (9 punti)
Si consideri un file system dove la dimensione di un blocco logico è pari a 4KB e con indirizzi di blocchi a 64 bit. Determinare la dimensione massima di un file in caso di
1. allocazione concatenata;
2. allocazione indicizzata con due livelli di blocco indice.

Determinare inoltre il numero di blocchi effettivamente assegnati ai dati di un file di 900KB per i due tipi di allocazione.

$DB = 4KB = 2^{12} B$ $\quad$ $\#puntatori = \frac{4KB}{8B} = 512$

$IB = 64 bit = 8B$

$DIM\ MAX\ FILE\ (concatenata) = 2^{66}(2^{12} - 8) = 2^{76} - 2^{67} = 64 ZB - 128 EB$

$DIM\ MAX\ FILE\ (indicizzata) = 512 \cdot 512 \cdot 4KB = 1GB$

$\#blocchi\ concatenata = \frac{900KB}{4KB - 8B} = 226$ $\quad$ $\#blocchi\ indicizzata = \frac{900KB}{4KB} = 225$

Esercizio 4 (6 punti)
La File Allocation Table è un esempio di allocazione concatenata per i file. Si determini la dimensione in byte di una FAT nei casi di:
1) Un vecchio floppy-disk da 1.44MB che usa blocchi da un KB;
2) Un disco da 3TB che usa blocchi da 4MB.

Quali sono i principali vantaggi dell'uso della FAT rispetto all'allocazione concatenata "classica"?

1) $N_{FAT} = \frac{1,44 MB}{KB} = 1475 \Rightarrow 2^x = 1475$ $\quad$ $x = 11\ (bit) = 1,375B \Rightarrow DIM\ FAT = 1475 \cdot 1,375B = 2028 B$

rcizio 4 (8 punti)

Si consideri un file system allocato su un disco con blocchi logici e fisici di 1024 byte. Supponiamo che le informazioni sulla localizzazione del file a cui vogliamo accedere siano contenute in memoria centrale. Per ciascuna delle tre strategie di allocazione dei file su disco — contigua, concatenata e con blocco indice — si risponda alle seguenti domande:

1. Come avviene la traduzione da indirizzo logico (riferito all'inizio del file) a indirizzo fisico (dato da numero di blocco del disco e offset all'interno del blocco)?
2. Se l'ultimo blocco del file a cui è stato fatto accesso è il blocco (logico) 8, ed ora si vuole accedere al blocco (logico) 6, quanti blocchi fisici devono essere letti dal disco?

(Per l'allocazione con blocco indice, supponiamo che il file sia completamente indirizzabile attraverso i puntatori diretti contenuti nel blocco stesso previamente caricato in memoria.)

Esercizio 3 (8 punti)

Si consideri un file system realizzato utilizzando i-node, ciascuno dei quali occupa un blocco, di dimensione pari a 512 byte. Ogni i-node contiene 12 puntatori diretti a blocchi di dati e 2 puntatori a blocchi indice, a singolo e doppio livello di indirezione. Sapendo che gli indirizzi di blocco sono espressi su 32 bit, si vuole allocare un file composto da 8000 record da 60 byte ciascuno, in modo tale che un record non possa essere suddiviso su due blocchi.

1) Calcolare quanti blocchi verranno utilizzati per allocare i dati del file;
2) Determinare l'occupazione totale di memoria secondaria relativa al file e risultante dalla strategia di allocazione descritta.

12 punt diretti $\quad D_B = 512\ B$
1 " singola
1 " doppia $\quad I_B = 32\ bit = 4B$

a) 8000 record da 60B

$$\#\text{punt per blocco} = \frac{512\ B}{4\ B} = 128 \qquad \#\text{record per blocco} = \frac{512\ B}{60\ B} = 8 \qquad \#\text{blocchi occupati} = \frac{8000}{8} = 1000$$

b) 12 al nodo principale
$1000 - 12 = 988$ → 1 blocco

128 con singola
$988 - 128 = 860$ → 1 blocco

doppia $\frac{860}{128} = 7$ blocchi

$\rightsquigarrow$ 10 blocchi per gestire l'allocazione

$$\text{occupazione memoria secondaria} = (1000 + 10)\ 512\ B = 517.120\ B$$