SISTEMI OPERATIVI, CORSI A e B Prova del 26/5/2009 CORSO |A| |B|

COGNOME E NOME ..... MATRICOLA ..... AULA FILA POSTO

ESERCIZIO 1 (4 punti)

In un sistema operativo che gestisce la memoria con rilocazione dinamica e caricamento in partizioni variabili, la memoria fisica ha un'ampiezza di 40 Mbyte. La partizione 1, della lunghezza di 5 Mbyte, è riservata al sistema operativo, mentre il resto della memoria è disponibile per il caricamento dei processi.
Quando viene generato un nuovo processo, il sistema operativo individua una partizione libera adatta a contenere il processo con la tecnica del best fit. Se non trova aree di memoria libere di dimensione sufficiente, il sistema operativo effettua lo swap-out dei processi caricati in memoria a partire dai processi più grandi (a parità di lunghezza si procede in ordine alfabetico), finchè non si libera un'area di memoria sufficiente a contenere il processo appena generato.
I processi che hanno subito uno swap-out vengono caricati in memoria (swap-in) non appena si libera un'area sufficiente a contenerli in seguito alla terminazione di qualche altro processo. Lo swap-in avviene in ordine rigorosamente FIFO (i primi processi che hanno subito lo swap-out sono i primi ad ottenere lo swap-in).
Al tempo t sono caricati in memoria i seguenti processi:

- Processo A, nella partizione con origine 5 Mbyte e lunghezza 6 Mbyte;
- Processo B, nella partizione con origine 16 Mbyte e lunghezza 2 Mbyte;
- Processo C, nella partizione con origine 21 Mbyte e lunghezza 4 Mbyte;
- Processo D, nella partizione con origine 29 Mbyte e lunghezza 3 Mbyte;
- Processo E, nella partizione con origine 34 Mbyte e lunghezza 5 Mbyte;

Successivamente si verificano le seguenti serie di eventi (in alternativa):

Serie 1:

Al tempo t+1 viene generato il processo F il cui spazio virtuale occupa 3 MB
Al tempo t+2 viene generato il processo G il cui spazio virtuale occupa 7 MB
Al tempo t+3 termina il processo B

Serie 2:

Al tempo t+1 viene generato il processo F il cui spazio virtuale occupa 9 MB
Al tempo t+2 viene generato il processo G il cui spazio virtuale occupa 3 MB
Al tempo t+3 termina il processo B

Mostrare come varia l'allocazione di memoria agli istanti t+1, t+2 e t+3 in ciascuna delle due serie di eventi.

SOLUZIONE

Al tempo t:

A	A	A	A	A	A						B	B				C	C	C	C					D	D	D			E	E	E	E	E	
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

Serie 1)

Al tempo t+1:

A	A	A	A	A	A						B	B	F	F	F	C	C	C	C					D	D	D			E	E	E	E	E	
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

Al tempo t+2:

G	G	G	G	G	G	G					B	B	F	F	F	C	C	C	C					D	D	D			E	E	E	E	E	
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

Al tempo t+3:

G	G	G	G	G	G	G	A	A	A	A	A	A	F	F	F	C	C	C	C					D	D	D			E	E	E	E	E	
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

Serie 2)

Al tempo t+1:

F	F	F	F	F	F	F	F	F			B	B				C	C	C	C					D	D	D			E	E	E	E	E	
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

Al tempo t+2:

F	F	F	F	F	F	F	F	F			B	B	G	G	G	C	C	C	C					D	D	D			E	E	E	E	E	
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

Al tempo t+3:

F	F	F	F	F	F	F	F	F					G	G	G	C	C	C	C					D	D	D			E	E	E	E	E	
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

ESERCIZIO 2 (4 punti)

Un disco con 2 facce, 1000 settori per traccia e 2000 cilindri ha un tempo di seek proporzionale al numero di cilindri attraversati e pari a 0,01 millisec per ogni cilindro. Il periodo di rotazione è di 0,2 millisec: conseguentemente il tempo impiegato per percorrere un settore è di 0,2 microsec.
A un certo tempo (convenzionalmente indicato come t=0) termina l'esecuzione dei comandi sul cilindro 800 e sono pervenute, nell'ordine, le seguenti richieste di lettura o scrittura:

- clindro 66, faccia 0, settore 80
- clindro 801, faccia 1, settore 988

Sucessivamente arrivano tre ulteriori comandi di lettura:

- clindro 166, faccia 0, settore 165 al tempo 7,2
- clindro 50, faccia 1, settore 676 al tempo 7,7
- clindro 1902, faccia 1, settore 309 al tempo 25
- clindro 777, faccia 1, settore 876 al tempo 20

Calcolare il tempo necessario per eseguire tutte queste operazioni supponendo che si adotti la politica di scheduling SCAN, e che la direzione della testina al tempo 0 sia verso il basso (verso i cilindri di indice minore).
Il tempo di esecuzione di ogni operazione è uguale alla somma dell'eventuale tempo di *seek,* del ritardo rotazionale (tempo necessario per raggiungere il settore indirizzato) e del tempo di percorrenza del settore indirizzato. Per il ritardo rotazionale dopo un'operazione di *seek* si assume sempre il valore di caso peggiore, pari a un intero periodo di rotazione.

SOLUZIONE

op. su cilindro:	66	settore:	80						
inizio:	0	seek:	7,34	rotazione:	0,2	percorrenza:	0,0002	fine:	7,5402
op. su cilindro:	166	settore:	165						
inizio:	7,5402	seek:	1	rotazione:	0,2	percorrenza:	0,0002	fine:	8,7404
op. su cilindro:	801	settore:	988						
inizio:	8,7404	seek:	6,35	rotazione:	0,2	percorrenza:	0,0002	fine:	15,2906
op. su cilindro:	50	settore:	676						
inizio:	15,2906	seek:	7,51	rotazione:	0,2	percorrenza:	0,0002	fine:	23,0008
op. su cilindro:	777	settore:	876						
inizio:	23,0008	seek:	7,27	rotazione:	0,2	percorrenza:	0,0002	fine:	30,471
op. su cilindro:	1902	settore:	309						
inizio:	30,471	seek:	11,25	rotazione:	0,2	percorrenza:	0,0002	fine:	41,9212

ESERCIZIO 3 (4 punti)

In un file system UNIX i blocchi del disco hanno ampiezza di 1Kbyte e i puntatori ai blocchi sono a 32 bit. Gli i-node contengono, oltre agli altri attributi, 5 indirizzi diretti e 3 indirizzi indiretti.
Si consideri il file rappresentato dal seguente i-node:

ind	0	1	2	3	4	5	6	7
Blocco fisico	981	766	9018	4232	5625	661	662	12091

Alcuni frammenti dei blocchi 301, 303, 661, 662, 5625 e 12091 sono riportati nel seguito.

Blocco fisico 301:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	100	101	102	103	104	105	106	107	108	109	…

Blocco fisico 303:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	6100	6101	6102	6103	6104	6105	6106	6107	6108	6109	…

Blocco fisico 661:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	735	736	737	663	664	665	800	801	802	803	…

Blocco fisico 662:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	300	301	302	303	304	305	306	307	308	309	…

Blocco fisico 5625:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	…

Blocco fisico 12091:

Indice nel blocco	0	1	2	3	4	5	6	7	8	9	…
Blocco fisico	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	…

Dire in quali blocchi fisici del disco sono contenuti i seguenti byte del file:

- Byte numero 5100
- Byte numero 9987
- Byte numero 12290
- Byte numero 6329
- Byte numero 10800
- Byte numero 15359

SOLUZIONE

Indirizzo	BLOCCO LOGICO	BLOCCO FISICO	Raggiunto attraverso:			
			INDIRIZZO DIRETTO	BLOCCO INDIRETTO PRIMO LIVELLO	BLOCCO INDIRETTO SECONDO LIVELLO	BLOCCO INDIRETTO TERZO LIVELLO
5100	4	5625	4	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
9987	9	664		Blocco 661 Ind. nel blocco 4	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
12290	12	801		Blocco 661 Ind. nel blocco 7	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
6329	6	736		Blocco 661 Ind. nel blocco 1	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
10800	10	665		Blocco 661 Ind. nel blocco 5	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....
15359	14	803		Blocco 661 Ind. nel blocco 9	Blocco ..... Ind. nel blocco.....	Blocco ..... Ind. nel blocco.....

ESERCIZIO 4 (4 punti)

In un sistema che gestisce la memoria con paginazione a domanda, le pagine logiche e i blocchi fisici hanno una lunghezza di 2^{12} byte e gli indirizzi logici hanno una lunghezza di 32 bit.

Per la gestione della memoria si utilizzano tabelle delle pagine a 2 livelli. Sia la tabella di primo livello che quelle di secondo livello contengono 2^{10} elementi di 4 byte: pertanto ogni tabella occupa una pagina e la componente dell'indirizzo logico che individua le pagine è suddivisa in due parti di 10 bit ciascuna, denominate *Ind_1°_Livello* e *Ind_2°_Livello*.

La tabella di primo livello è caricata permanentemente in memoria principale, in un blocco noto al sistema operativo, mentre le tabelle di secondo livello sono caricate a domanda.

La tabella di primo livello è indicizzata dall'indice *Ind_1°_Livello,* che è anche l'indice di una tabella di secondo livello. L'elemento *Tabella_1°_Livello[Ind_1°_Livello]* contiene, tra l'altro, l'indicatore *P* di presenza della tabella di secondo livello di indice *Ind_1°_Livello* ed eventualmente l'indice del blocco fisico nel quale la tabella medesima è caricata. Ogni tabella di secondo livello è indicizzata dall'indice *Ind_2°_Livello,*e l'elemento *Tabella_2°_Livello[Ind_2°_Livello]* contiene, tra l'altro, l'indicatore *P* di presenza della pagina ed eventualmente l'indice del blocco fisico nel quale la pagina è caricata.

Al tempo t è in esecuzione il processo P e i contenuti parziali della sua tabella delle pagine di primo livello e di alcune tabelle di secondo livello sono mostrati in figura. Nella memoria fisica sono disponibili alcuni blocchi, che al verificarsi di *page faults* sono utilizzabili per caricare tabelle di secondo livello o pagine del processo. Questi blocchi sono ordinati nella coda <PrimoBlocco> → 40 → 41 → 42 → 43 → 44 → 45…….. e, in caso di *page fault*, sono utilizzati in questo ordine.

Indice 1° Liv	Blocco	P
0000000000	2345	1
0000000001	--	0
0000000010	2234	1
0000000011	3456	1
0000000100	--	0
0000000101	--	0
0000000110	--	0
0000000111	3567	1
0000001000	3678	1
0000001001	3890	1
0000001010	--	0
0000001011	--	0
0000001100	--	0
0000001101	--	0
0000001110	3901	1
…….		

Tabella 1° Livello

Indice 2° Liv	Blocco	P
0000000000	8000	1
0000000001	--	0
0000000010	--	0
0000000011	--	0
0000000100	8901	1
0000000101	8012	1
0000000110	8123	1
0000000111	--	0
0000001000	--	0
0000001001	--	0
0000001010	--	0
0000001011	8765	1
0000001100	--	0
0000001101	--	0
0000001110	8567	1
…….		

Tabella 2° Livello Indice 0000000100

Indice 2° Liv	Bl	P
0000000000	--	0
0000000001	--	0
0000000010	--	0
0000000011	7890	1
0000000100	7901	1
0000000101	7012	1
0000000110	--	0
0000000111	--	0
0000001000	--	0
0000001001	--	0
0000001010	7123	1
0000001011	--	0
0000001100	--	0
0000001101	7234	1
0000001110	7345	1
…….		

Tabella 2° Livello Indice 0000000111

Indice 2° Liv	Blocco	P
0000000000	--	0
0000000001	--	0
0000000010	--	0
0000000011	--	0
0000000100	--	0
0000000101	--	0
0000000110	9001	1
0000000111	9001	1
0000001000	9101	1
0000001001	--	0
0000001010	--	0
0000001011	--	0
0000001100	--	0
0000001101	9102	1
0000001110	9221	1
…….		

Tabella 2° Livello Indice 0000000010

Indice 2° Liv	Blocco	P
0000000000	9500	1
0000000001	--	0
0000000010	--	0
0000000011	--	0
0000000100	9511	1
0000000101	9520	1
0000000110	9520	1
0000000111	--	0
0000001000	--	0
0000001001	--	0
0000001010	--	0
0000001011	9533	1
0000001100	--	0
0000001101	--	0
0000001110	9639	1
…….		

Tabella 2° Livello Indice 0000001010

Indice 2° Liv	Blocco	P
0000000000	9789	1
0000000001	--	0
0000000010	--	0
0000000011	--	0
0000000100	--	0
0000000101	9776	1
0000000110	--	0
0000000111	--	0
0000001000	9765	1
0000001001	--	0
0000001010	--	0
0000001011	--	0
0000001100	9754	1
0000001101	9743	1
0000001110	--	0
…….		

Tabella 2° Livello Indice 0000000000

A partire dal tempo t il processo P accede alla memoria con i seguenti indirizzi binari, nei quali è omessa la componente di 12 bit riservata all'offset nella pagina:

1. indirizzo 0000000111 0000001000
2. indirizzo 0000001010 0000001001
3. indirizzo 0000000100 0000001110
4. indirizzo 0000000000 0000001100

Per ciascun accesso alla memoria, si chiede:

- se l'accesso alla tabella di secondo livello determina page fault
- il blocco fisico che contiene (eventualmente dopo il caricamento in memoria) la tabella di secondo livello
- se l'accesso alla pagina riferita determina page fault
- blocco fisico che contiene (eventualmente dopo il caricamento in memoria) la pagina riferita.

SOLUZIONE

1) Indirizzo 0000000111 0000001000

l'accesso alla tabella di 2° livello determina page fault?	NO
blocco fisico che contiene la tabella di secondo livello	3567
l'accesso alla pagina riferita determina page fault?	SI
blocco fisico che contiene la pagina riferita	40

2) Indirizzo 0000001010 0000001001

l'accesso alla tabella di 2° livello determina page fault?	SI
blocco fisico che contiene la tabella di secondo livello	41
l'accesso alla pagina riferita determina page fault?	SI
blocco fisico che contiene la pagina riferita	42

3) Indirizzo 0000000100 0000001110

l'accesso alla tabella di 2° livello determina page fault?	SI
blocco fisico che contiene la tabella di secondo livello	43
l'accesso alla pagina riferita determina page fault?	NO
blocco fisico che contiene la pagina riferita	8567

4) Indirizzo 0000000000 0000001100

l'accesso alla tabella di 2° livello determina page fault?	NO
blocco fisico che contiene la tabella di secondo livello	2345
l'accesso alla pagina riferita determina page fault?	NO
blocco fisico che contiene la pagina riferita	9754

ESERCIZIO 5 (4 PUNTI)

Un sistema operativo che gestisce la memoria con paginazione a domanda con pagine (e blocchi fisici) di 2 Kbyte, è dotato di un File System FAT 16 con le seguenti caratteristiche:

- il File System è ospitato da un disco della capacità di 64 Mbyte, con blocchi di 2 Kbyte;
- gli elementi della FAT hanno una lunghezza di 2 byte;
- la FAT risiede stabilmente sul disco ed è logicamente suddivisa in pagine di 2 Kbyte, che vengono caricate in memoria a domanda.

Lo stato di occupazione della memoria è descritto dalla *Core Map,* i cui elementi hanno i campi *Id* (indice di un processo o identificatore della FAT, o *null* se il blocco è libero) *Pag* (pagina del processo o della *FAT* caricata nel blocco), *Rif* (indicatore di pagina riferita) e *Mod* (indicatore di modifica). L'algoritmo di sostituzione utilizzato è il *Second Chance.*
Al tempo *t* sono caricate in memoria pagine dei processi A, B, C e della FAT, e la *CoreMap* ha la configurazione mostrata in figura. Ad esempio il blocco 10 è occupato dalla pagina 1 della *FAT,* con indicatore di riferimento uguale a 0 e indicatore di modifica uguale a 0. I primi 6 blocchi della memoria fisica sono riservati al sistema operativo e sono ignorati dall'algoritmo di sostituzione. Al tempo *t* il puntatore dell'algoritmo *Second Chance* è posizionato sul blocco 11.
Al tempo *t* è in esecuzione il processo A, che esegue una chiamata di sistema per leggere i blocchi logici 2 e 3 del file *filename.* Il blocco logico 2 di *filename* risiede nel blocco fisico 2500 del disco, e il contenuto dell'elemento di indice 2500 della FAT è 10000. La chiamata di sistema esegue codice del nucleo, che non riferisce pagine soggette a caricamento dinamico, eccetto quelle della *FAT.*

Id							C	A	B	A	FAT	B	C	A	C	FAT	C	B	A	B	FAT	C	FAT	C
Pag							0	1	0	2	1	6	3	8	9	5	12	2	7	3	20	7	58	2
Rif							0	1	0	0	0	1	1	0	1	1	0	0	1	0	1	1	0	0
Mod							0	1	1	0	0	0	1	1	1	0	0	0	0	1	0	1	1	1
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Core Map al tempo *t*

Si chiede:

1) il numero di elementi della FAT;
2) il numero di byte e il numero di pagine che compongono la FAT;
3) i blocchi fisici del disco che ospitano i blocchi logici 2 e 3 di *filename;*
4) quali pagine della FAT vengono riferite per leggere i blocchi logici 2 e 3 di *filename;*
5) la configurazione della *Core Map* dopo la lettura del blocco logico 2 e del blocco logico 3 di *filename;*
6) il numero di accessi al disco che vengono eseguiti per trasferire in memoria i due blocchi logici.

SOLUZIONE

1) numero di elementi della FAT: uguale al numero di blocchi del disco → $2^{26} / 2^{11} = 2^{15}$ elementi;
2) numero di byte e il numero di pagine che compongono la FAT: $2^{15} * 2 = 2^{16}$ byte; $2^{16} / 2^{11} = 32$ pagine
3) il blocco logico 2 risiede nel blocco fisico 2500;
   il blocco logico 3 risiede nel blocco fisico FAT[2500]= 10000
4) per leggere il blocco logico 2 viene riferita la pagina 2500 **div** $2^{11} = 1$ della FAT
   per leggere il blocco logico 3 viene riferita la pagina 10000 **div** $2^{11} = 4$ della FAT
5) configurazione della *Core Map* dopo la lettura del blocco logico 2:

Id							C	A	B	A	FAT	B	C	A	C	FAT	C	B	A	B	FAT	C	FAT	C
Pag							0	1	0	2	1	6	3	8	9	5	12	2	7	3	20	7	18	2
Rif							0	1	0	0	1	1	1	0	1	1	0	0	1	0	1	1	0	0
Mod							0	1	1	0	0	0	1	1	1	0	0	0	0	1	0	1	1	1
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

   configurazione della *Core Map* dopo la lettura del blocco logico 3:

Id							C	A	B	A	FAT	B	C	FAT	C	FAT	C	B	A	B	FAT	C	FAT	C
Pag							0	1	0	2	1	6	3	4	9	5	12	2	7	3	20	7	58	2
Rif							0	1	0	0	1	0	0	1	1	1	0	0	1	0	1	1	0	0
Mod							0	1	1	0	0	0	1	0	1	0	0	0	0	1	0	1	1	1
Blocco	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

6) numero di accessi al disco che vengono eseguiti per trasferire in memoria i due blocchi: 4
   Precisamente:
   - 1 accesso per salvare sul disco la pagina 8 del processo A
   - 1 accesso per caricare da disco la pagina 4 della FAT
   - 1 accesso per trasferire in memoria il blocco fisico 2500 di *filename.*
   - 1 accesso per trasferire in memoria il blocco fisico 10000 di *filename.*

ESERCIZIO 6 (2 punti)

In un sistema UNIX che adotta la segmentazione, i segmenti sono caricati in memoria fisica con la tecnica delle partizioni variabili e con politica *first fit.* Ad un dato istante di tempo lo spazio logico del processo P comprende i segmenti *codice*, *dati* e *pila* allocati nelle partizioni di origine 5600, 45900 e 31000 e lunghezze 8000, 14000 e 3000, rispettivamente.
Successivamente il processo P esegue una *fork* che genera il processo F. Al momento della *fork* sono libere le partizioni *Part1* con origine 15000 e lunghezza 15000, *Part 2* con origine 36000 e lunghezza 5000 e *Part3* con origine 370000 e lunghezza 20000. Si chiedono i contenuti dei registri base e limite dei segmenti codice, dati e pila quando è in esecuzione il processo P e quando è in esecuzione il processo F.

SOLUZIONE

1) Quando è in esecuzione il processo P:

Segmento codice:	registro base 5600	registro limite 8000
Segmento dati:	registro base 45900	registro limite 14000
Segmento pila:	registro base 31000	registro limite 3000

2) Quando è in esecuzione il processo F:

Segmento codice:	registro base 5600	registro limite 8000
Segmento dati:	registro base 15000	registro limite 14000
Segmento pila:	registro base 36000	registro limite 3000

ESERCIZIO 7 (2 punti)

In un sistema UNIX, la memoria virtuale di ogni processo comprende i segmenti *codice*, *dati* e *pila.* Il caricamento nella memoria fisica avviene con la tecnica delle partizioni variabili.
Il processo A, che occupa le partizioni di origine 21000, 40000 e 60000 e lunghezze 10000, 7000 e 4000 rispettivamente per i segmenti *codice, dati* e *pila*, esegue la chiamata di sistema *exec.* I segmenti *codice1, dati1* e *pila1* che sostituiscono quelli originari hanno rispettivamente lunghezze 12000, 20000 e 2000.
Al momento della chiamata sono libere le seguenti partizioni;

- inizio 31.000, lunghezza 9.000;
- inizio 47.000, lunghezza 13.000;
- inizio 64.000, lunghezza 6.000;
- inizio 80.000, lunghezza 40.000.

Il gestore della memoria adotta la politica *best fit* e rilascia le partizioni inizialmente occupate dai segmenti *codice, dati* e *pila*, del processo A solo dopo aver eseguito l'assegnazione per i segmenti *codice1, dati1* e *pila1*.
Si chiede:

1. l'origine e la lunghezza delle partizioni assegnate ai segmenti *codice1, dati1* e *pila1 ;*
2. l'origine e la lunghezza delle partizioni che rimangono libere dopo questa assegnazione e dopo il successivo rilascio delle partizioni assegnate ai segmenti *codice, dati* e *pila*.

SOLUZIONE

1) Partizione assegnata al segmento *codice1:* origine 47.000, lunghezza 12.000
Partizione assegnata al segmento *dati1:* origine 80.000, lunghezza 20.000
Partizione assegnata al segmento *pila1:* origine 64.000 lunghezza 2.000
2) Partizioni che rimangono libere dopo l'assegnazione e dopo il successivo rilascio delle partizioni assegnate ai segmenti *codice, dati* e *pila*

- origine 21.000, lunghezza 26.000
- origine 59.000, lunghezza 5.000
- origine 66.000, lunghezza 4.000
- origine 100.000, lunghezza 20.000

ESERCIZIO 8 (2 punti)

In un file system UNIX i blocchi del disco hanno ampiezza di 1Kbyte e gli i-node contengono 10 indirizzi diretti e 3 indirizzi indiretti. Tutti gli indirizzi hanno una lunghezza di 4 byte.
Si chiede:

1. la massima capacità del disco che ospita il file system, in blocchi e in byte
2. la massima dimensione dei file indirizzabili dall'i-node, in blocchi e in byte.

SOLUZIONE

1. Massima capacità del disco che ospita il file system: 2^{32} blocchi; $2^{32}*2^{10}= 2^{42}$ byte (16 Tbyte)
2. considerato che:

   - lo i-node indirizza direttamente 10 blocchi
   - il blocco indiretto semplice indirizza 2^{10} **div** 4 = 2^8 blocchi dati
   - il blocco indiretto di secondo livello indirizza 2^{10} **div** 4 =2^8 blocchi indiretti di primo livello, ciascuno dei quali indirizza 2^8 blocchi dati,
   - il blocco indiretto di terzo livello indirizza 2^{10} **div** 4 =2^8 blocchi indiretti di secondo livello, ciascuno dei quali indirizza 2^{10} **div** 4 =2^8 blocchi indiretti di primo livello, ciascuno dei quali indirizza 2^8 blocchi dati,

   la massima dimensione dei file indirizzabili dall'i-node è pari a:

   - $10 + 2^8 + 2^{16} + 2^{24}= 10+ 256+ 65536+ 16777216= =16.843.018$ blocchi,
   - ovvero 16.843.018 Kbyte.

ESERCIZIO 9 (2 PUNTI)

Un disco è organizzato con *NCilindri= 200* (numerati da 0 a 199), *NFacce= 4* (numerate da 0 a 3) e *NSettori= 1000* (numero di settori per traccia; numerati da 0 a 999). Ogni settore contiene 2^8= 256 byte, pari a un blocco.
A livello logico i blocchi sono individuati con *indici di blocco,* interi compresi nell'intervallo *[0, MaxBlocchi),* e sono allocati sul disco secondo la sequenza *cilindro, faccia, settore* (ad esempio, il blocco 1900 corrisponde alla terna *(c= 0, f= 1, s= 900),* dove *c* è l'indice di cilindro, *f* è l'indice di faccia e *s* è l'indice di settore.
Si chiede:

1) la capacità del disco, in numero di blocchi e di byte
2) l'indice di blocco corrispondente alle terne:
   - *(c= 22, f= 2, s= 0)*;
   - *(c= 17, f= 2, s= 899)*;
3) la terna corrispondente i blocchi:
   - 189000
   - 1000

SOLUZIONE

- Ogni cilindro contiene *NFacce*NSettori*= 4* 1000 = 4000 blocchi;

la capacità del disco è : 4000 * 200= 800.000 blocchi, pari a 800.000*256 bytes = 200.000 Kbyte

- Dalla formula: *b= c*(NFacce*NSettori) + f*NSettori + s ,*

l'indice di blocco corrispondente alla terna *(c= 22, f= 2, s= 0)* è 90.000

l'indice di blocco corrispondente alla terna *(c= 17, f= 2, s=899)* è 70.899

- Dalle formule:

  c= b div *(NFacce*NSettori);*

  f= (b mod *(NFacce*NSettori))* div *NSettori;*

  s= (b mod *(NFacce*NSettori))* mod *NSettori* ,

la terna *(c, f, s)* corrispondente all'indice 189000 è *(c= 47, f= 1, s= 0)*.

la terna *(c, f, s)* corrispondente all'indice 1000 è *(c= 0, f= 1, s= 0)*.

ESERCIZIO 10 (2 punti)

In un file system UNIX dove ogni i-node occupa 1 blocco, si consideri il file */usr/carlo/documenti/compito.* Calcolare il numero di accessi al disco necessari per aprire questo file, supponendo che ogni cartella di questo path occupi 1 blocco e che lo i-node della cartella radice sia caricato in memoria, mentre tutti gli altri i-node e tutte le cartelle interessate risiedono su disco

SOLUZIONE

1. 1 accesso per leggere la cartella /
2. 1 accesso per leggere lo i-node della cartella *usr*
3. 1 accesso per leggere la cartella *usr*
4. 1 accesso per leggere lo i-node della cartella *carlo*
5. 1 accesso per leggere la cartella *carlo*
6. 1 accesso per leggere lo i-node della cartella *documenti*
7. 1 accesso per leggere la cartella *documenti*
8. 1 accesso per leggere lo i-node del file *compito* e caricarlo nella tabella dei file aperti.

In totale : **8** accessi.