

ESERCIZIO 1 (4 PUNTI)

In un sistema che gestisce la memoria con partizioni variabili, il sistema operativo occupa una partizione con origine 0 e lunghezza 6. Al tempo 0 sono stati generati e caricati in memoria i seguenti processi:

- il processo A, che risiede in memoria da 6 sec ed è caricato nella partizione con origine 6 e lunghezza 5;
- il processo B, che risiede in memoria da 4 sec ed è caricato nella partizione con origine 13 e lunghezza 7;
- il processo C, che risiede in memoria da 2 sec ed è caricato nella partizione con origine 25 e lunghezza 3;
- il processo D, che risiede in memoria da 1 sec ed è caricato nella partizione con origine 28 e lunghezza 8;

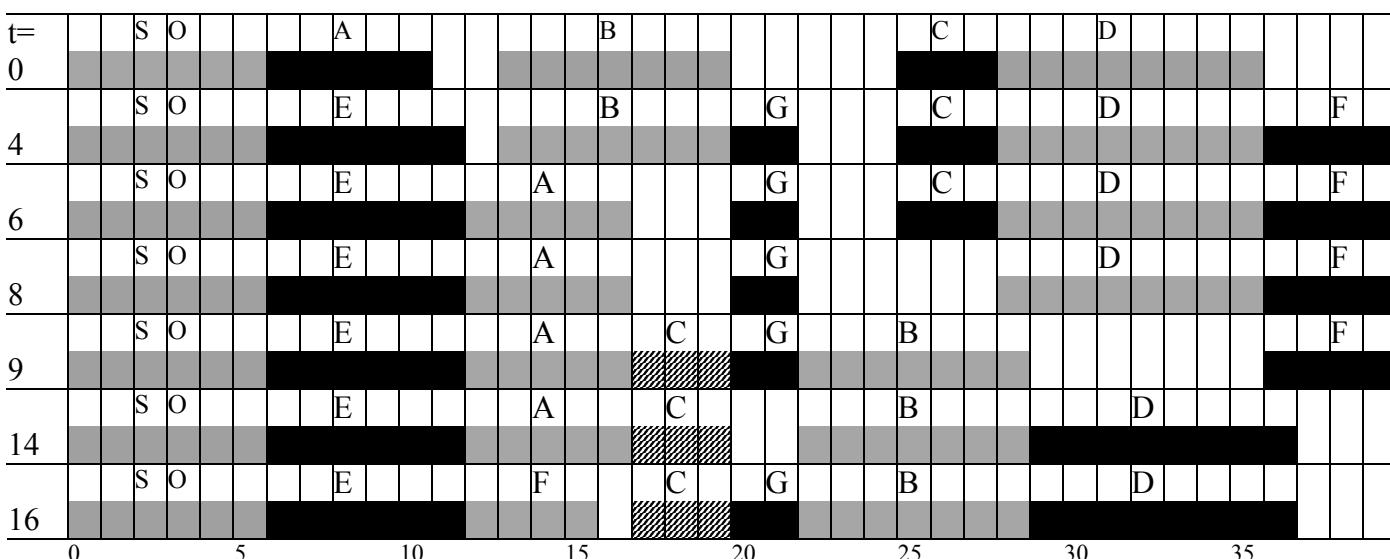
Inoltre sono stati generati i seguenti processi, che al tempo 0 sono in attesa di caricamento:

- il processo E, che richiede una partizione di lunghezza 6;
- il processo F, che richiede una partizione di lunghezza 4;
- il processo G, che richiede una partizione di lunghezza 2.

Il codice dei processi è rilocabile. I processi in attesa di caricamento sono inseriti in una coda *FIFO*.

A ogni processo caricato in memoria viene revocata l'assegnazione quando sono trascorsi 10 sec dal caricamento. In questa circostanza interviene lo scheduler il quale carica, se possibile, uno o più processi in attesa di caricamento, adottando la politica FIFO per la scelta dei processi da caricare e la politica *best-fit* per la scelta delle partizioni da assegnare.

Utilizzando il grafico sotto riportato, mostrare come evolvono l'occupazione della memoria e la coda dei processi in attesa di caricamento fino al tempo 16.

RISPOSTA

	PROCESSI							
	A (5)	B(7)	C(3)	D(8)	E(6)	F(4)	G(2)	CODA
t= 0	SI, 4	SI, 6	SI, 8	SI, 9	NO	NO	NO	E → F → G
t= 4	NO	SI, 6	SI, 8	SI, 9	SI, 14	SI, 14	SI, 14	A
t= 6	SI, 16	NO	SI, 8	SI, 9	SI, 14	SI, 14	SI, 14	B
t= 8	SI, 16	NO	NO	SI, 9	SI, 14	SI, 14	SI, 14	B → C
t= 9	SI, 16	SI, 19	SI, 19	NO	SI, 14	SI, 14	SI, 14	D
t= 14	SI, 16	SI, 19	SI, 19	SI, 24	SI, 24	NO	NO	F → G
t= 16	NO	SI, 19	SI, 19	SI, 24	SI, 24	SI, 26	SI, 26	A

Nota: al tempo t= 14 vengono scaricati i processi E, F, G. Il comportamento successivo dipende dall'ordine con il quale vengono inseriti nella coda prima di procedere al caricamento. Nella soluzione si è supposto l'ordinamento D → E → F → G, che riflette l'ordine di generazione, ma nella correzione sono state accettati anche ordinamenti diversi.

ESERCIZIO 2 (4 PUNTI)

In un sistema che gestisce la memoria con paginazione a domanda sono presenti i processi A, B, C, D.

Lo stato di occupazione della memoria al tempo 20 è descritto dalla tabella *Core Map*. Gli elementi di questa tabella (in corrispondenza con i blocchi fisici) hanno i campi *Proc* (processo a cui è assegnato il blocco; il campo è vuoto se il blocco è libero); *Pag* (pagina del processo caricata nel blocco), *t* (tempo dell'ultimo riferimento alla pagina.) I primi 6 blocchi sono riservati al sistema operativo.

Le tabelle delle pagine dei processi A e B sono mostrate in figura.

Per la gestione della memoria si utilizza un algoritmo di sostituzione LRU locale. Il *working set* assegnato al processo A (inteso come numero di blocchi a disposizione del processo, anche se momentaneamente non occupati) ha dimensione 5; quello assegnato al processo B ha dimensione 3.

Proc						C	A	B	A		B	C	D		A	D	B	A		C		C		
Pag						0	1	0	2		6	3	1		5	6	2	7		7		2		
t						6	2	1	5		10	13	9		12	19	15	18		8		16		
	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 20

Pagina	Blocco
0	-
1	7
2	9
3	-
4	-
5	15
6	-
7	18
Proc. A	t=20

Pagina	Blocco
0	8
1	-
2	17
3	-
4	-
5	-
6	11
7	-
Proc. B	t=20

Il processo A riferisce la pagina 1 al tempo 21, la pagina 6 al tempo 22, la pagina 0 al tempo 23 e la pagina 5 al tempo 24;

Il processo B riferisce la pagina 0 al tempo 25 e la pagina 4 al tempo 26.

Come si modificano i contenuti della *Core Map* e delle tabelle delle pagine dei processi A e B ai tempi 21, 22, 23, 24, 25 e 26 ?

RISPOSTA

Core Map:

t= 21 elemento 7 nuovi valori dei campi: Proc =A; Pag =1 ; t =21 ;
t= 22 elemento 10 nuovi valori dei campi: Proc =A ; Pag =6 ; t = 22 ;
t= 23 elemento 9 nuovi valori dei campi: Proc =A ; Pag =0 ; t =23 ;
t= 24 elemento 15 nuovi valori dei campi: Proc =A ; Pag =5 ; t = 24 ;
t= 25 elemento 8 nuovi valori dei campi: Proc =B ; Pag =0 ; t = 25 ;
t= 26 elemento 11 nuovi valori dei campi: Proc =B ; Pag =4 ; t = 26 ;

Proc						C	A	B	A	A	B	C	D		A	D	B	A		C		C		
Pag						0	1	0	0	6	4	3	1		5	6	2	7		7		2		
t						6	21	25	23	22	26	13	9		24	19	15	18		8		16		
	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 26

Tabelle delle pagine

Pagina	Blocco
0	-
1	7
2	9
3	-
4	-
5	15
6	-
7	18
Proc. A	t=21

Pagina	Blocco
0	-
1	7
2	9
3	-
4	-
5	15
6	10
7	18
Proc. A	t=22

Pagina	Blocco
0	9
1	7
2	-
3	-
4	-
5	15
6	10
7	18
Proc. A	t=23

Pagina	Blocco
0	9
1	7
2	-
3	-
4	-
5	15
6	10
7	18
Proc. A	t=24

Pagina	Blocco
0	8
1	-
2	17
3	-
4	-
5	-
6	11
7	-
Proc. B	t=25

Pagina	Blocco
0	8
1	-
2	17
3	-
4	11
5	-
6	-
7	-
Proc. B	t=26

COGNOME NOME MATRICOLA

ESERCIZIO 3 (4 PUNTI)

Un disco con 4 facce, 30 settori per traccia e 120 cilindri ha un tempo di seek (proporzionale al numero di cilindri attraversati) pari a 0,5 ms per ogni cilindro. Il periodo di rotazione è di 12 msec: di conseguenza il tempo impiegato per percorrere un settore è 0,4 msec.

Al tempo 0 termina l'esecuzione dei comandi relativi al cilindro 65 e sono pendenti le seguenti richieste di lettura o scrittura:

- cilindro 5: settore 10 della faccia 0;
- cilindro 35: settore 5 della faccia 2 e settore 25 della faccia 3
- cilindro 100: settore 6 della faccia 1 e settore 6 della faccia 2;

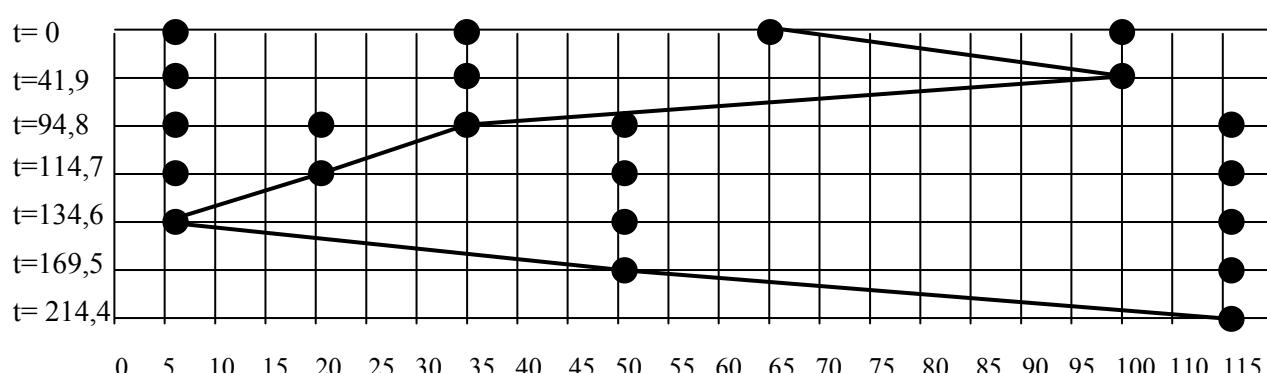
Successivamente arrivano i seguenti comandi:

- al tempo 50: settore 15 della faccia 0 del cilindro 50.
- al tempo 70: settore 12 della faccia 1 del cilindro 115
- al tempo 80: settore 15 della faccia 2 del cilindro 20.

Lo scheduling è effettuato con politica SCAN. La fase attiva al tempo 0 è quella di salita.

Calcolare il tempo necessario per eseguire tutte le operazioni. 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. Quando si raggiunge un cilindro, i comandi pendenti devono essere eseguiti nell'ordine in cui sono elencati.

Per il ritardo rotazionale dopo un'operazione di seek si assume sempre il valore di caso peggiore, pari a un periodo di rotazione.



RISPOSTA

- 1) Operazione sul cilindro 100

Inizio t= 0 tempo di seek 0,5*(100-65) ritardo rotazionale 12 tempo di percorrenza 0,4 fine t= **29,9**

- 2) Operazione sul cilindro 100

Inizio t= 29,9 tempo di seek 0 ritardo rotazionale 12-0,4 tempo di percorrenza 0,4 fine t= **41,9**

- 3) Operazione sul cilindro 35

Inizio t= 41,9 tempo di seek 0,5*(100-35) ritardo rotazionale 12 tempo di percorrenza 0,4 fine t= **86,8**

Sono arrivati i comandi sui cilindri 50, 115, 20

- 4) Operazione sul cilindro 35

Inizio t= 86,8 tempo di seek 0 ritardo rotazionale 19*0,4 tempo di percorrenza 0,4 fine t= **94,8**

- 5) Operazione sul cilindro 20

Inizio t= 94,8 tempo di seek 15*0,5 ritardo rotazionale 12 tempo di percorrenza 0,4 fine t= **114,7**

- 6) Operazione sul cilindro 5

Inizio t= 114,7 tempo di seek 15*0,5 ritardo rotazionale 12 tempo di percorrenza 0,4 fine t= **134,6**

- 7) Operazione sul cilindro 50

Inizio t= 134,6 tempo di seek 45*0,5 ritardo rotazionale 12 tempo di percorrenza 0,4 fine t= **169,5**

- 8) Operazione sul cilindro 115

Inizio t= 169,5 tempo di seek 65*0,5 ritardo rotazionale 12-0,4 tempo di percorrenza 0,4 fine t= **214,4**

COGNOME NOME MATRICOLA

ESERCIZIO 4 (4 PUNTI)

Un disco RAID di livello 4 è composto da 5 dischi fisici, numerati da 0 a 4. I blocchi del disco virtuale V sono mappati nei dischi 0, 1, 2, 3: precisamente il blocco b del disco V è mappato nel blocco $b \bmod 4$ del disco fisico di indice b . Il disco 4 è ridondante e il suo blocco di indice i contiene la parità dei blocchi di indice i dei dischi 0, 1, 2, 3.

Il gestore del disco virtuale accetta comandi (di lettura o scrittura) che interessano più blocchi: ad esempio $read(buffer, b_1, b_2, b_3)$, dove, b_1, b_2, b_3 sono blocchi del disco virtuale.

A un certo tempo viene eseguita l'operazione $read(buffer, 12, 13, 14)$, dove i blocchi 12, 13, 14 del disco virtuale sono mappati nel blocco 3 dei dischi fisici 0, 1, 2. Supponiamo che la lettura dal disco fisico 1 fallisca e che questo evento venga riconosciuto e segnalato dal gestore. Si chiede:

1. Quale altro blocco si deve leggere per ricostruire il contenuto del blocco 3 del disco fisico 1 ?
2. Qual è il contenuto ricostruito del blocco 3 del disco fisico 1 se i blocchi di indice 3 dei dischi fisici 0, 2, 3, 4 contengono 0100 1101, 01101001, 01111001, 11101100 ?
3. Se successivamente si esegue una scrittura sul blocco 3 del disco fisico 1, il cui contenuto diviene 10011011, come deve essere modificato il blocco 3 del disco fisico 4 ?

RISPOSTA

1. Blocco di indice 3 del disco fisico 3 e 4
2. Contenuto ricostruito del blocco 3 del disco fisico 1:

Disco 0	0	1	0	0	1	1	0	1
Disco 1	1	0	1	1	0	0	0	1
Disco 2	0	1	1	0	1	0	0	1
Disco 3	0	1	1	1	1	0	0	1
Disco 4	1	1	1	0	1	1	0	0

3. Contenuto modificato del blocco 3 del disco fisico 4:

Disco 0	0	1	0	0	1	1	0	1
Disco 1	1	0	0	1	1	0	1	1
Disco 2	0	1	1	0	1	0	0	1
Disco 3	0	1	1	1	1	0	0	1
Disco 4	1	1	0	0	0	1	1	0

ESERCIZIO 5 (4 PUNTI)

In un file system UNIX dove gli *i-node* occupano 1 blocco e contengono 10 indirizzi diretti e 3 indirizzi indiretti, si consideri il file *esempio*, che è già aperto. Si esegue un'operazione di lettura, che interessa i blocchi logici 8, 9, 10, 11, 12 del file (i blocchi logici sono numerati a partire da 0). Tutti i blocchi dati, compresi i blocchi degli indirizzi indiretti, risiedono su disco. Gli indirizzi contenuti nello *i-node* e nel blocco indiretto semplice sono mostrati nelle tabelle.

Quali accessi al disco sono necessari per eseguire l'operazione?

Indirizzi nello <i>i-node</i>	0	1	2	3	4	5	6	7	8	9	10	11	12
Valore	25	27	41	54	55	67	68	80	81	83	90	91	95

Indirizzi nel blocco indiretto semplice	0	1	2	3	4	5	6
Valore	29	30	42	64	65	66	78

RISPOSTA

Vengono letti dal disco i blocchi:

1. blocco 81, che contiene **blocco logico 8**
2. blocco 83, che contiene **blocco logico 9**
3. blocco 90, che contiene **blocco indiretto semplice**
4. blocco 29, che contiene **blocco logico 10**
5. blocco 30, che contiene **blocco logico 11**
6. blocco 42, che contiene **blocco logico 12**

COGNOME NOME MATRICOLA

ESERCIZIO 6 (2 PUNTI)

Un sistema che gestisce la memoria con paginazione a domanda utilizza l'algoritmo di sostituzione *Second Chance* (globale). Sono presenti i processi A, B, C, D. Lo stato della memoria fisica è descritto dalla seguente *Core Map*, dove ogni elemento ha campi *Proc* (processo a cui è assegnato il blocco); *Pag* (pagina del processo caricata nel blocco), *r* (bit di pagina riferita). Si trascurano i blocchi assegnati al sistema operativo.

Proc	A	A	B	B	D	C	C	A	B	A	C	B	C	D	B	A	D	B	A	B	A	C	D	C
Pag	3	4	1	4	0	4	0	1	0	2	5	6	3	1	8	5	6	2	7	9	8	7	3	2
r	1	1	1	1	0	0	1	1	0	1	1	0	0	1	1	1	0	0	1	1	1	1	1	0
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

Al tempo t il puntatore è posizionato sul blocco 6 e il processo D riferisce la pagina 2. Al tempo t+1 continua ad avanzare il processo D che riferisce la pagina 8. Quale è la posizione finale del puntatore e come si modifica la core map?

Nota: dopo ogni intervento, il puntatore si posiziona sul blocco successivo a quello che conteneva la vittima.

RISPOSTA

1. Posizione del puntatore dopo l'intervento di *Second Chance* al tempo t : **9**
2. Posizione finale del puntatore (dopo l'intervento di *Second Chance* al tempo t+1): **12**
3. Configurazione finale della *Core Map* (correggere I valori che si modificano):

Proc	A	A	B	B	D	C	C	A	D	A	C	D	C	D	B	A	D	B	A	B	A	C	D	C
Pag	3	4	1	4	0	4	0	1	2	2	5	8	3	1	8	5	6	2	7	9	8	7	3	2
r	1	1	1	1	0	0	0	0	1	0	0	1	0	1	1	1	0	0	1	1	1	1	1	0
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+1

ESERCIZIO 7 (2 PUNTI)

Un sistema UNIX gestisce la memoria con segmentazione combinata con paginazione. La memoria virtuale di ogni processo comprende i segmenti *codice* e *dati*, quest'ultimo comprensivo della pila. Per la gestione si utilizzano, oltre alla *Core Map*, anche due tabelle delle pagine per ogni processo, rispettivamente per il segmento codice e per il segnato dati.

Il processo *ProcK* esegue la chiamata di sistema *fork* che genera il processo *ProcJ*. Al momento della chiamata la tabelle delle pagine del segmento codice e del segmento dati di *ProcK* hanno i seguenti contenuti:

Pagina	Blocco
0	-
1	7
2	9
3	-
4	-
5	15
6	-
7	18

ProcK: segmento codice

Pagina	Blocco
0	-
1	-
2	17
3	-
4	-
5	-
6	11
7	8

ProcK: segmento dati

La primitiva che esegue la chiamata di sistema crea sul disco un file di swap per il processo *ProcJ* e vi copia i dati del padre, quindi assegna e inizializza le tabelle delle pagine di *ProcJ*, lasciando al meccanismo della paginazione a domanda il compito di caricare in memoria i dati. Quel è il contenuto delle tabelle delle pagine del processo *ProcJ* subito dopo la *fork* ?

RISPOSTA

Pagina	Blocco
0	-
1	7
2	9
3	-
4	-
5	15
6	-
7	18

ProcJ: segmento codice

Pagina	Blocco
0	-
1	-
2	-
3	-
4	-
5	-
6	-
7	-

ProcJ: segmento dati

COGNOMENOMEMATRICOLA

ESERCIZIO 8 (2 PUNTI)

Si consideri un sistema dove gli indirizzi logici hanno la lunghezza di 32 bit e le pagine logiche e fisiche hanno ampiezza di 2 kByte. Per la gestione della memoria con paginazione a domanda si utilizzano tabelle delle pagine a 2 livelli. La tabella di primo livello comprende 2^{10} elementi.

Gli elementi di ogni tabella di primo o secondo livello occupano 3 byte, all'interno dei quali 2 bit sono riservati agli indicatori di pagina caricata e di pagina riferita, mentre i rimanenti codificano un indice di blocco fisico.

Si chiede:

1. la lunghezza del campo offset, in numero di bit;
2. la lunghezza delle tabelle di secondo livello (numero di elementi);
3. lo spazio occupato in memoria dalla tabella di primo livello (numero di byte);
4. lo spazio occupato in memoria da ogni tabella di secondo livello (numero di byte);
5. la massima dimensione della memoria fisica, in numero di blocchi e di byte.

RISPOSTA

1. Lunghezza del campo offset: **11** bit;
2. Lunghezza delle tabelle di secondo livello: **$2^{11} = 2048$ elementi**)
3. Spazio occupato in memoria dalla tabella di primo livello: **$3 \cdot 2^{10} = 3\text{Kbyte}$** ;
4. Spazio occupato in memoria da ogni tabella di secondo livello: **$3 \cdot 2^{11} = 6\text{Kbyte}$** ;
5. Massima dimensione della memoria fisica **2^{22} blocchi; $2^{22} \cdot 2^{11} = 8\text{Gbyte}$** .

ESERCIZIO 9 (2 PUNTI)

In un file system FAT è definito il file *esercizi/FileSystem*. Il contenuto parziale dell'elemento della directory *esercizi* che lo individua è (*FileSystem*, 24).

Il disco che ospita il file system ha 30 blocchi. Il file *FileSystem* occupa 10 blocchi, individuati dalla FAT, il cui contenuto parziale (per la parte che descrive questo file) è il seguente:

		11	5	6	16				12	28				Ø																	
0	1	2	3	4	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		

La FAT risiede su disco, dove occupa 3 blocchi, che contengono rispettivamente gli elementi (0 .. 9), (10 .. 19) e (20 .. 29) della FAT. Il gestore del disco ha un buffer di un solo blocco e pertanto la lettura di un blocco distrugge la copia in memoria del blocco letto precedentemente.

Per leggere il blocco logico 8 del file partendo dall'elemento (*FileSystem*, 24) della directory, si deve percorrere la lista definita sulla FAT fino a caricare il memoria il blocco del disco che contiene la posizione su disco del blocco richiesto, e quindi leggere il blocco medesimo. Quanti accessi al disco sono necessari per completare l'operazione?

RISPOSTA)

Accessi per leggere i seguenti blocchi:

1. blocco che contiene gli elementi **20-29** della FAT
2. blocco che contiene gli elementi **0-9** della FAT
3. blocco che contiene gli elementi **10-19** della FAT
4. blocco che contiene gli elementi **20-29** della FAT
5. blocco che contiene gli elementi **0-9** della FAT
6. 1 accesso per leggere il blocco dati richiesto

ESERCIZIO 10 (2 PUNTI)

In un file system UNIX dove ogni i-node occupa 1 blocco, si consideri il file */usr/tizio/public_html/index.htm*

Supponendo che ogni directory di questo path occupi 1 blocco e che lo i-node della directory radice sia caricato in memoria, mentre tutti gli altri i-node e tutte le directory interessate risiedono su disco, calcolare il numero di accessi al disco necessari per leggere il primo blocco del file considerato.

RISPOSTA

1. 1 accesso per leggere **directory /**
2. 1 accesso per leggere **i-node usr**
3. 1 accesso per leggere **directory usr**
4. 1 accesso per leggere **i-node tizio**
5. 1 accesso per leggere **directory tizio**
6. 1 accesso per leggere **i-node public_html**
7. 1 accesso per leggere **directory public_html**
8. 1 accesso per leggere **i-node index.htm**
9. 1 accesso per leggere **primo blocco index.htm**

In totale : **9** accessi.