POLITECNICO DI TORINO

Programmazione di sistema

Esame del 07/07/2023 (OS internals - Cabodi)

Iniziato	venerdì, 7 luglio 2023, 14:04
Terminato	venerdì, 7 luglio 2023, 15:24
Tempo impiegato	1 ora 20 min.
Valutazione	**13,90** su un massimo di 15,00 (**93**%)
Feedback	Attenzione. La valutazione finale è saturata a 13. Eventuali voti > 13 sono pertanto portati automaticamente a 13.

Domanda 1

Completo

Punteggio ottenuto 3,00 su 3,00

Si consideri la seguente stringa di riferimenti di memoria, per un determinato processo. Per ogni riferimento (Byte addressing, con indirizzi espressi in esadecimale) viene riportata anche l'operazione di lettura/scrittura (R/W): R 33FB, R 1B64, W 30D3, W 237E, R 0AC8, W 23D7, R 174A, R 0965, W 32A0, R 1BB0, W 09E5, R 3380, R 2A94, R 11B8. Si supponga che gli indirizzi fisici e logici siano su 16 bit, la dimensione della pagina sia 2KByte e 70FF sia l'indirizzo massimo utilizzabile dal programma (il limite superiore dello spazio degli indirizzi).

A) Calcolare la dimensione dello spazio degli indirizzi (address space, espresso come numero di pagine) e la frammentazione interna.

B) Calcolare la stringa di riferimenti di pagina.

C) Simulare un algoritmo di sostituzione di pagine di tipo **second-chance**, con 3 frame disponibili. Rappresentare il resident set (frame fisici contenenti pagine logiche) dopo ogni riferimento in memoria. Indicare esplicitamente, per ogni frame allocato, il numero di pagina e il bit di riferimento (utilizzare la notazione p,r o p_r). Indicare anche i page fault Si supponga che il bit di riferimento di una pagina sia inizializzato a 0 dopo il relativo page fault.

A) Calcolare la dimensione dello spazio degli indirizzi (address space, espresso come numero di pagine) e la frammentazione interna.

Pagine da 2 KB = 2^11 B

Indirizzi su 16 bit --> I primi (16-11=) 5 bit rappresentano l'indice di pagina

Indirizzo massimo 0x70FF = 0111 0,000 1111 1111

L'indice di pagina massimo è rappresentato dai primi 5 bit dell'indirizzo massimo --> 01110 = 14

Siccome gli indici vengono numerati a partire da 0, lo spazio di indirizzamento contiene 15 pagine

L'ultima pagina è occupata fino all'offset 000 1111 1111 = 255

La framm interna sarà quindi 2 KB - (255 + 1) B = 1792 B

B) Calcolare la stringa di riferimenti di pagina.

Siccome l'indice di pagina è rappresentato dai primi 5 bit dell'indirizzo, per ottenerlo basta moltiplicare per 2 la prima cifra esadecimale e sommare a questo numero il MSB (bit più significativo) della seconda cifra hex.

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

C) Simulare un algoritmo di sostituzione di pagine di tipo **second-chance**, con 3 frame disponibili. Rappresentare il resident set (frame fisici contenenti pagine logiche) dopo ogni riferimento in memoria. Indicare esplicitamente, per ogni frame allocato, il numero di pagina e il bit di riferimento (utilizzare la notazione p,r o p_r). Indicare anche i page fault Si supponga che il bit di riferimento di una pagina sia inizializzato a 0 dopo il relativo page fault.

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13
References	6	3	6	4	1	4	2	1	6	3	1	6	5	2
Resident Set	6,0	6,0	6,1	6,1	6,0	6,0	2,0	2,0	2,0	3,0	3,0	3,0	5,0	5,0
		3,0	3,0	3,0	1,0	1,0	1,0	1,1	1,0	1,0	1,1	1,1	1,0	2,0
				4,0	4,0	4,1	4,0	4,0	6,0	6,0	6,0	6,1	6,0	6,0
Page Faults	x	x		x	x		x		x	x			x	x

Commento:
ok

Domanda 2

Completo

Punteggio ottenuto 2,50 su 3,00

Si consideri un'unità disco rigido (HDD) con tecnologia e geometria standard. Il disco ha

- dimensione del settore di 512 Byte,
- 2048 tracce per faccia,
- 50 settori per traccia,
- 5 dischi/piatti a doppia faccia,
- Tempo medio di seek di 10 msec.

A) Qual è la capacità di una traccia (quanti Byte può contenere)? Qual è la capacità di ogni faccia? Qual è la capacità del disco? Quanti cilindri ha il disco?

B) Il metodo di organizzazione di disco CHR (Cylinder-Head-Sector) consente a un blocco del disco di estendersi su più settori di una data traccia (identificata da una coppia cilindro, testina) ma non su tracce diverse. Le seguenti dimensioni dei blocchi sono valide? 256 B, 2048 B, 51200 B (rispondere per ognuna e motivare le risposte)

C) Se i dischi ruotano a 5400 giri / min (giri al minuto), qual è la latenza rotazionale massima? Qual è la latenza rotazionale media? Si sa che è possibile trasferire un'intera traccia di dati per rtazione, qual è la velocità di trasferimento (espressa in bit/secondo)? Quale sarebbe la velocità di trasferimento se un intero cilindro di dati potesse essere trasferito in una rotazione?

A) Qual è la capacità di una traccia (quanti Byte può contenere)? Qual è la capacità di ogni faccia? Qual è la capacità del disco? Quanti cilindri ha il disco?

|Settore| = 512 B

|Traccia| = 50 * |Settore| = 25 KB

|Faccia| = 2048 * |Traccia| = 50 MB

|Disco| = 5 * 2 * |Faccia| = 500 MB

Considerando che ci sono 2048 tracce per ogni faccia, il disco conterrà 2048 cilindri.

B) Il metodo di organizzazione di disco CHR (Cylinder-Head-Sector) consente a un blocco del disco di estendersi su più settori di una data traccia (identificata da una coppia cilindro, testina) ma non su tracce diverse. Le seguenti dimensioni dei blocchi sono valide? 256 B, 2048 B, 51200 B (rispondere per ognuna e motivare le risposte)

La dimensione del blocco diviso per la dimensione del settore (512 B) non deve eccedere i 50 (settori per ogni traccia, per non occupare più di una traccia). Quindi le dimensioni 256 B e 2048 B vanno bene (occupano meno di 50 settori), mentre la dimensione 51200 B risulta eccessiva (si occuperebbero 12800 settoril

C) Se i dischi ruotano a 5400 giri / min (giri al minuto), qual è la latenza rotazionale massima? Qual è la latenza rotazionale media? Si sa che è possibile trasferire un'intera traccia di dati per rtazione, qual è la velocità di trasferimento (espressa in bit/secondo)? Quale sarebbe la velocità di trasferimento se un intero cilindro di dati potesse essere trasferito in una rotazione?

5400 rpm / 60 = 90 giri/s

Lat. = 1 / 90 = ~11.1 ms

Lat._avg = 1/2 * Lat. = ~5.55 ms

|Traccia| = 25 KB

v = |Traccia| / Lat._avg = 25 KB / 5.55 ms = 25 K * 8 bit / (5.55*10^-3) s = ~35.2 Mbit/s

|Cilindro| = |Traccia| * 10 (facce) = 250 KB

v = |Cilindro| / Lat._avg = 250 KB / 5.55 ms = 250 K * 8 bit / (5.55*10^-3) = ~351,9 Mbit/s

Commento:

A) ok -> 1

B) non considera il vincolo di multiplo di settore -> 0,75

C) ok ma sbaglia a conteggiare il tempo per mezza traccia nei calcoli di velocità -> 0,75

Domanda 3

Completo

Punteggio ottenuto 2,40 su 3,00

Un'applicazione deve organizzare un file con un'intestazione (header) e più sezioni. L'header include, tra gli altri dati, offset e dimensioni di tutte le sezioni del file.

A) Per ciascuna delle domande seguenti, rispondere sì / no e motivare la risposta

		Yes/No	Motivate/explain
1	L'applicazione può supporre che l'header, così come ogni sezione inizi all'inizio di un blocco del disco (nel file system)?		
2	Può L'applicazione decidere di organizzare i dati in più file (anziché in un singolo file): un file per l'intestazione e uno per ogni sezione? (Se NO, motivare, se SÌ, motivare e spiegare come gli offset di sezione nell'intestazione devono essere rappresentati, codificati o modificati)		

B) Un file di testo viene copiato in tre diversi file system, con formati di allocazione file, rispettivamente:

- allocazione contigua (per multiplo di un blocco di disco)
- allocazione collegata (senza FAT)
- allocazione basata s Inode

Rispondere alla seguente domanda e motivare la risposta:

		Yes/No	Motivate/explain
1	La dimensione di un blocco su disco è la stessa in tutti e tre i file system. Possiamo supporre che la frammentazione interna sarà la stessa in tutti i file system?		

C) Un file system contiene tre file di testo a.txt, b.txt e c.txt. Un'applicazione di archiviazione/compressione di file (come ad esempio gzip, tar, 7z, rar, ecc.) memorizza il (contenuto dei) tre file di testo all'interno di un singolo file di archivio (ad esempio abc.zip).

Per ciascuna delle domande seguenti, rispondere sì / no e motivare la risposta

		Yes/No	Motivate/explain
1	I tre file di testo possono condividere lo stesso blocco del disco, il che significa che il file system li memorizza (almeno parzialmente) all'interno dello stesso blocco del disco?		

2	È possibile, per il contenuto dei tre file di testo, replicati e/o compressi nel file di archivio, condividere (almeno parzialmente) lo stesso blocco disco (in modo da essere memorizzato all'interno dello stesso blocco disco)?		

A) Per ciascuna delle domande seguenti, rispondere sì / no e motivare la risposta

		Yes/No	Motivate/explain
1	L'applicazione può supporre che l'header, così come ogni sezione inizi all'inizio di un blocco del disco (nel file system)?	No	Non necessariamente una sezione deve iniziare all'inizio di un blocco del disco: la memorizzazione all'interno dei blocchi è gestita ad un livello diverso rispetto all'organizzazione del file.
2	Può L'applicazione decidere di organizzare i dati in più file (anziché in un singolo file): un file per l'intestazione e uno per ogni sezione? (Se NO, motivare, se SÌ, motivare e spiegare come gli offset di sezione nell'intestazione devono essere rappresentati, codificati o modificati)	No	No, in quanto l'organizzazione della memoria non è gestita direttamente dall'applicazione.

B) Un file di testo viene copiato in tre diversi file system, con formati di allocazione file, rispettivamente:

- allocazione contigua (per multiplo di un blocco di disco)
- allocazione collegata (senza FAT)
- allocazione basata s Inode

Rispondere alla seguente domanda e motivare la risposta:

		Yes/No	Motivate/explain
1	La dimensione di un blocco su disco è la stessa in tutti e tre i file system. Possiamo supporre che la frammentazione interna sarà la stessa in tutti i file system?	No	Per quanto sia vero che la dimensione del file sia la stessa in tutti e tre i file system (e quindi sia indipendente da essi), la sua occupazione all'interno di un file system dipende dall'organizzazione del file system stesso, e dunque non possiamo affermare che la frammentazione interna sia la stessa in tutti e tre i casi.

C) Un file system contiene tre file di testo a.txt, b.txt e c.txt. Un'applicazione di archiviazione/compressione di file (come ad esempio gzip, tar, 7z, rar, ecc.) memorizza il (contenuto dei) tre file di testo all'interno di un singolo file di archivio (ad esempio abc.zip).

Per ciascuna delle domande seguenti, rispondere sì / no e motivare la risposta

		Yes/No	Motivate/explain
1	I tre file di testo possono condividere lo stesso blocco del disco, il che significa che il file system li memorizza (almeno parzialmente) all'interno dello stesso blocco del disco?	No	No, in quanto la memorizzazione dei diversi file prevede che questi occupino blocchi diversi in memoria.
2	È possibile, per il contenuto dei tre file di testo, replicati e/o compressi nel file di archivio, condividere (almeno parzialmente) lo stesso blocco disco (in modo da essere memorizzato all'interno dello stesso blocco disco)?	Sì	Sì, è possibile che dopo l'operazione di compressione il contenuto dei file sia condiviso in uno stesso blocco del disco, anche solo parzialmente, proprio perché l'archiviazione punta ad ottenere il minor spazio occupato possibile.

Commento:

A1) ok -> 0,6

A2) Ma la domanda non è sull'organizzazione della memoria o dei file, E' solo su come un'applicazione gestisca i "suoi" dati-> 0

B) ok -> 0,6

C1) ok -> 0,6

C2) ok -> 0,6

Domanda 4

Completo

Punteggio ottenuto 3,00 su 3,00

E' dato un sistema OS161. Si consideri il framework del file system utilizzato in LAB5, parzialmente rappresentato dai seguenti frammenti di codice:

Code excerpts from files proc.h and file_syscalls.c

```
struct proc {
        ...
#if OPT_FILE
        /* per-process open file table */
        struct openfile *fileTable[OPEN_MAX];
#endif
};
```

```
/* system open file table */
struct openfile {
  struct vnode *vn;
  off_t offset;
  unsigned int countRef;
};
struct openfile systemFileTable[SYSTEM_OPEN_MAX];
```

Rispondere alle seguenti domande:

A) Perché systemFileTable è un array di struct openfile, mentre fileTable è un array di puntatori a struct openfile?

B) Il campo countRef è ridondante, considerando che il vnode puntato dal campo vn ha (internamente) il suo contatore di riferimenti?

C) Nel caso in cui sia necessario supportare operazioni di lock dei file, cosa andrebbe scelto tra uno spinlock e un lock e dove dovrebbe essere posizionato (o posizionati, se multipli): in systemFIleTable, in fileTable o altrove?

D) Qual è il possibile ruolo delle funzioni copyin e copyout nell'implementazione delle chiamate di sistema sys_read e sys_write?

A) Perché systemFileTable è un array di struct openfile, mentre fileTable è un array di puntatori a struct openfile?

Perché la systemFileTable è una tabella a livello di sistema, comune a tutti i processi, che tiene traccia dei file aperti (attraverso le struct openfile). Una fileTable, invece, è una tabella per ogni processo (un vettore di puntatori a struct openfile), le cui entry puntano alle struct openfile presenti nella systemFileTable.

B) Il campo countRef è ridondante, considerando che il vnode puntato dal campo vn ha (internamente) il suo contatore di riferimenti?

No, non è ridondante, in quanto possono esserci più livelli di condivisione: essendoci un vnode per ogni file aperto, un vnode può essere puntato da più entry della systemFileTable (in questo modo si giustifica la presenza del contatore interno al vnode), mentre, come detto in precedenza, una entry della systemFileTable può essere puntata da più fileTable, e dunque essere comune a più processi (da qui la necessità di due contatori dei riferimenti separati).

C) Nel caso in cui sia necessario supportare operazioni di lock dei file, cosa andrebbe scelto tra uno spinlock e un lock e dove dovrebbe essere posizionato (o posizionati, se multipli): in systemFIleTable, in fileTable o altrove?

Per garantire l'accesso in mutua esclusione su di un file potrebbe essere utilizzato un lock (evitando il busy waiting derivante dallo spinlock), inserito come un campo della struct vnode.

D) Qual è il possibile ruolo delle funzioni copyin e copyout nell'implementazione delle chiamate di sistema sys_read e sys_write?

Le funzioni copyin e copyout effettuano la copia di aree di memoria (similmente a quando avviene con l'istruzione memmove), ma hanno come sorgente/destinazione un pezzo di memoria kernel, e forniscono dei meccanismi di protezione aggiuntivi rispetto ad eventuali errori. La funzione copyin ha come destinazione memoria kernel, mentre la copyout ha come sorgente memoria kernel.

Commento:
ok

Domanda 5

Completo

Punteggio ottenuto 3,00 su 3,00

Si considerino, in OS161, le due possibili implementazioni (parziali) di spinlock_acquire mostrate di seguito:

```
void spinlock_acquire(struct spinlock *splk) {
  // ...
  while (1) {
    if (spinlock_data_get(&splk->splk_lock) != 0) {
      continue;
    }
    if (spinlock_data_testandset(&splk->splk_lock) != 0) {
      continue;
    }
    break;
  }
  // ...
}
```

```
void spinlock_acquire2(struct spinlock *splk) {
  // ...
  while (1) {
    while (spinlock_data_get(&splk->splk_lock) != 0);
    if (spinlock_data_testandset(&splk->splk_lock) == 0) {
      continue;
    }
  }
  // ...
}
```

A) Perché spinlock_acquire utilizza sia spinlock_data_get che spinlock_data_testandset, invece di chiamarne solo una?

B) spinlock_acquire2 equivale a spinlock_acquire? Se sì, motivarlo, se no, motivarlo e, se possibile, modificarla per renderla equivalente

---

A) Perché spinlock_acquire utilizza sia spinlock_data_get che spinlock_data_testandset, invece di chiamarne solo uno?

Perché la funzione proposta ricorre alla tenica nota come test-and-test-and-set: prima di effettuare una test-and-set (per acquisire il possesso dello spinlock) effettua l'attesa (attende che lo spinlock si liberi) con una semplice operazione di lettura. Una volta che lo spinlock risulta libero, si procede con il tentativo di acquisizione tramite la funzione test-and-set. Questo pattern risulta più efficace rispetto alla sola test-and-set perché, come detto, l'attesa viene consumata attraverso un'operazione di lettura, che è meno dispendiosa rispetto ad una test-and-set.

B) spinlock_acquire2 equivale a spinlock_acquire? Se sì, motivarlo, se no, motivarlo e, se possibile, modificarla per renderla equivalente

Il comportamento risulterebbe equivalente alla spinlock_acquire se, nell'if di controllo per la funzione test_and_set, si controllasse che il valore sia diverso da 0. Nel caso il risultato della test_and_set==0, infatti, si è acquisito il controllo dello spinlock, e si può uscire dal ciclo di attesa while(1).

```
void spinlock_acquire2(struct spinlock *splk) {
  // ...
  while (1) {
    while (spinlock_data_get(&splk->splk_lock) != 0);
    if (spinlock_data_testandset(&splk->splk_lock) != 0) {
      continue;
    }
    break;
  }
  // ...
}
```

Commento:
ok

Domanda 6

Risposta non data

Non valutata

Per ritirarti, seleziona "Mi ritiro".

In caso contrario, è equivalente non rispondere oppure rispondere " Desidero che Il mio esame sia valutato".

Potrai ancora comunicare l'intenzione di ritirarti a esame chiuso, una volta vista la soluzione proposta.

- (a) Mi ritiro (il mio esame non verrà valutato)
- (b) Desidero che Il mio esame sia valutato

Risposta errata.

La risposta corretta è: Mi ritiro (il mio esame non verrà valutato)