Sistemi Operativi — A.A. 2005-2006, prova scritta del 21 settembre 2006

Libri e appunti chiusi. Vietato comunicare con chiunque. Vietato l'uso di cellulari, calcolatrici, palmari e affini.
Tempo a disposizione: 60 minuti. Le domande sono etichettate con 1,2 o 3 asterischi:
* = domanda semplice, valutazione alta, rispondi a queste prima delle altre
** = domanda di media difficoltà
*** = domanda difficile, valutazione bassa, rispondi dopo aver risposto alle altre

1. * Mostra lo schema che permette di tradurre indirizzi virtuali in reali per mezzo di inverted page tables. Descrivi i campi dell'inverted page table. Descrivi il problema delle collisioni e la sua soluzione.

schema	campi dell'inverted page table
vedi materiale didattico	Il numero del frame è il valore hash *f* calcolato, tale informazione non è quindi tra i campi della tabella. L'accesso alla inverted page table viene fatto per verificare che *f* sia un valore corretto. Per ogni riga *f* della IPT (e quindi per ogni frame *f*) i campi sono • process id del processo *p* a cui il frame *f* è assegnato • numero della pagina di *p* contenuta in frame *f* • chain: in caso di collisioni si scorre una lista di frames • control bits
Il problema delle collisioni vedi materiale didattico	

2. * Indica se le seguenti affermazioni sono vere o false con una crocetta nella rispettiva colonna.

Domanda	**vero**	**falso**
Una system call dà sempre luogo ad un mode switch.	x	
Un process switch avviene sempre contestualmente a 2 mode switch.	x	
Un interrupt viene gestito in modalità utente.		x
Il process switch può avvenire sia in modalità kernel che in modalità utente.		x
Il dispatcher viene sempre eseguito contestualmente ad un mode switch di tipo kernel→ user.	x	
Se lo scheduling della CPU è preemptive, l'arrivo di dati genera sempre un process switch indipendentemente dalla priorità dei processi.		x
Se un processo è in blocco da 10ms significa che 10ms fa ha eseguito una system call	x	
Ogni interrupt può essere associato ad un processo che ha richiesto una operazione di I/O.		x
Molti page fault su un processo non modificano le prestazioni degli altri processi.		x
Un processo per ottenere nuova memoria deve fare una system call.	x	
Un processo per lanciare un nuovo processo deve fare una system call.	x	
Una system call bloccante causa sempre un process switch. (ambigua) vero se ci sono altri processi	x	

Sistemi Operativi — A.A. 2005-2006, prova scritta del 21 settembre 2006

3. ** Considera una architettura stile Pentium: pagina di 4 KB, paginazione a due livelli, pte 4 byte, root page table sempre in memoria. Il frammento di codice assembly mostrato è composto da 3 istruzioni che vengono eseguite consecutivamente. Calcola quanti page fault può generare al più ciascuna istruzione durante l'esecuzione del frammento in questione. Considera le istruzioni eseguite di seguito e supponi che le pagine caricate dalle istruzioni precedenti permangano residenti durante l'esecuzione delle istruzioni successive.

Indirizzo	istruzione	Page faults dovuti a parti di page table non residenti	Page faults dovuti a codice o dati non residenti
`0x003ffff6`	carica nel registro A 4 byte a partire da `0x803ffffe`	3	3
`0x003ffffB`	carica nel registro B 4 byte a partire da `0x80400ffe`	0	1
`0x00400000`	aggiungi ad A il valore di B	1	1

Pagine accedute dalle 3 istruzioni e relativi possibili faults

```
      codice          dati
    pt    page       pt   page      pt   page
1)  003   003ff      803  803ff     3    3
                     804  80400
2)  003   003ff      804  80400     0    1
                          80401
3)  004   00400      -              1    1
```

4. ** Considera un sistema con architettura del kernel "execution within user process". In tale sistema sono presenti tre processi: A e B sono I/O bound e C è puramente cpu-bound. Lo scheduler è round robin con quanto q. A è inizialmente in testa alla coda ready seguito da B e C. L'I/O burst di A dura $1.5q$ e quello di B dura $1.7q$. Il cpu burst di A, il cpu busrt di B, i tempi di dispatching e di esecuzione di system call e dell'interupt handler sono tutti molto piccoli e trascurabili rispetto a q.
Il processore esegue di volta in volta A, B, C, mode switching, dispatcing, system call e interrupt handler. Mostra schematicamente, nella seguente tabella, l'ordine con cui tali attività vengono eseguite (una sola croce per ciascuna colonna).

tempo →

user mode	A	x																									x
	B					x																					
	C									x				x				x				x					
mode switch			x				x				x				x		x		x		x		x			x	
kernel mode	dispatching				x				x				x													x		
	system call			x				x																				
	interrup handler											x				x				x				x				

A in blocco — B in blocco — t=q rr timer — t=1.5q input per A A ready — t=1.7q input per B B ready — t=2q rr timer

5. * Descrivi la tecnica di scheduling denominata feedback.

vedi materiale didattico

Cognome:_____ **Nome:** _____ **Matricola:**_____

Sistemi Operativi — A.A. 2005-2006, prova scritta del 21 settembre 2006

6. ** Considera i due sistemi raid4 e raid5 con 3+1 dischi. Supponi che ciascun disco impieghi un tempo *w* a scrivere un blocco. Assumi che il disk scheduler scriva sempre il blocco di parità contemporaneamente al blocco dati. Mostra i tempi che impiegano raid4 e raid5 nelle seguenti situazioni compilando la tabella.

Descrizione della situazione	***Tempo impiegato da raid4***	***Tempo impiegato da raid5***
Scrittura di un singolo blocco	*w*	*w*
Scrittura di *n* blocchi distribuiti uniformemente. Supponi che tutte le richieste siano note allo scheduler del disco all'istante iniziale e può quindi ottimizzare.	*nw*	*nw/2 (stima)*
Commento Raid4 deve scrivere ogni volta sul disco di parità, non può parallelizzare. Raid5 scrive ogni volta su due dischi. Poiché la distribuzione delle richieste è uniforme e le richieste sono tutte note in anticipo può scegliere sempre richieste con coppie di dischi non sovrapposte e quindi parallelizzabili. **Nota.** Sia per raid4 e che per raid5, per il calcolo della parità è necessario prima leggere sia il blocco *b* che si scrive sia la parità *p,* mediante xor si ottiene la parità *p'* di tutti i blocchi tranne *b*. Da *p'* si può ottenere la nuova parità. Nella tabella si assume che *w* sia la somma del tempo di lettura e di quello di scrittura. **Nota per il caso di *n* blocchi distribuiti uniformemente.** Se si assume che lo scheduler possa accorpare le scritture sia per raid4 che per raid5 il tempo di scrittura diviene *nw/3* poiché la parità viene scritta una sola volta e il collo di bottiglia sparisce.		

Considera ora raid5 con 3+1 dischi. Supponi che all'istante iniziale ci siano da schedulare 12 richieste per la scrittura dei blocchi 1,2,3,4,5,6,7,8,9,10,11,12. Mostra una possibile sequenza di scritture ottimizzata. Per chiarezza mostra anche l'organizzazione e la numerazione dei blocchi di raid5 con i blocchi di parità che stai considerando.

Numerazione blocchi raid 5

disk1	***disk2***	***disk3***	***disk4***
1	2	3	P
4	5	P	6
7	P	8	9
P	10	11	12
....			

Descrivi una sequenza di scritture ottimizzata dallo scheduler del disco.

Le scritture vengono parallelizzate a coppia
1,5
2,4
3,7
6,10
8,12
9,11
Se assumiamo che lo scheduler possa accorpare le richieste per il calcolo della parità allora abbiamo
1,2,3
4,5,6
7,8,9
10,11,12

7. ** Considera l'algoritmo di page replacement "aging" con 3 frame a disposizione, stimatore di anzianità a 3 bit con scorrimento a destra. Sweep ogni 4ms. Supponi che venga fatto un accesso a memoria ogni 1ms e la sequenza di accessi sia 1 2 3 1 2 2 2 2 3 3 3 3 3 2 1 1 4. Completa la seguente tabella.

Sistemi Operativi — A.A. 2005-2006, prova scritta del 21 settembre 2006

	0					4ms					8ms					12ms					16ms	
acc.	1	2	3	1		stimatori	2	2	2	2	stimatori	3	3	3	3	stimatori	3	2	1	1	stimatori	4
fault	f	f	f																			f
frame 1	1	1	1	1		'100	1	1	1	1	'010	1	1	1	1	'001	1	1	1	1	'100	4
frame 2		2	2	2		'100	2	2	2	2	'110	2	2	2	2	'011	2	2	2	2	'101	2
frame 3			3	3		'100	3	3	3	3	'010	3	3	3	3	'101	3	3	3	3	'110	3

Verifica se LRU sostituirebbe le stesse pagine ed eventualmente spiega il perché delle differenze.

LRU sostituirebbe la pagina 3 anziché la 1. Questo perché la granularità del campionamento (4ms) no permette di discriminare tra gli accessi avvenuti tra 12ms e 16ms che per aging sono sostanzialmente contemporanei.

8. ** Considera un sistema con scheduling round robin. Nel sistema sono presenti *n* processi I/O bound con cpu burst trascurabile. Quale frazione di tempo mediamente ciascun processo aspetterebbe in coda ready? (ignora il tempo di esecuzione del process switch).

0

Se i processi fossero tutti cpu bound, quale frazione di tempo ciascun processo aspetta in coda ready?

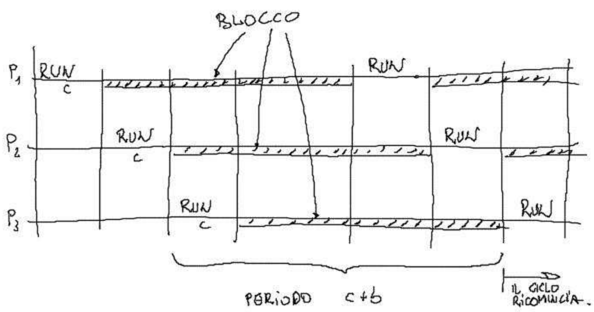
(n-1)/n.

Supponi ora di avere *n* processi tutti con I/O burst di durata *b* e cpu burst di durata *c* minore del quanto di tempo. Esprimi in formule la funzione *f(n,b,c)* che dà la frazione di tempo di cpu utilizzato (cioè con almeno un processo running). Giustifica la risposta.

$f(n,b,c)=1$ se $b \le (n-1)c$

$f(n,b,c)=nc/(b+c)$ se $b>(n-1)c$

Esempio con 3 processi:



9. *** Dai una classe di stringhe di riferimenti a memoria su cui l'algoritmo di page replacement FIFO da gli stessi page fault di CLOCK ma che non dia un fault ad ogni accesso. Considera 4 frame e 5 pagine.

varie soluzioni possibili ad esempio

1 2 3 4 (5$_{\text{esce 1}}$ 2 3 4 5 1$_{\text{esce 2}}$ 3 4 5 1 2$_{\text{esce 3}}$ 4 5 1 2 3$_{\text{esce 4}}$ 1 2 3 5 4$_{\text{esce 5}}$ 1 2 3 4)* *vedi materiale didattico per la spiegazione*

in alternativa (1 2 3 4 5 4 3)*

Cognome:_____ Nome: _____ Matricola:_____

Sistemi Operativi — A.A. 2005-2006, prova scritta del 21 settembre 2006