Cognome:_____ Nome: _____ Matricola:_____

Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 18 luglio 2005

Compito A

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

Domanda 1

1. * Che cosa è un mode switch? In quali occasioni si verifica?

Le moderne cpu possono lavorare in almeno due modalità: kernel mode, in cui si possono eseguire tutte le istruzioni macchina, user mode in cui solo alcune sono ammesse.
Si dice mode switch quella operazione che fa cambiare la cpu da uno stato ad un altro.
Avviene quando si serve un interrupt o una system call (user→kernel) o quando si fa il dispatching di un processo (kernel→user).

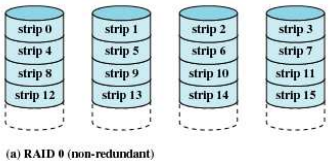
2. ** Considera un sistema con architettura "execution within user process" (come in UNIX) con scheduling non preemptive e *P* processi I/O bound che fanno, ciascuno, *N* system call di I/O bloccanti e una system call finale che termina il processo. Quanti mode switch sono generati fino al momento in cui l'ultimo processo termina? Quanti sarebbero i mode switch se le system call di I/O non fossero bloccanti? Giustificare le risposte.

Ciascun processo è una sequenza di N+1 cpu burst intervallati da N I/O burst generati dalle N system call di I/O. Si ha: avvio del primo cpu burst kernel→user, system call I/O user→kernel, ritorno da I/O kernel→user, terminazione user→kernel. Non vi sono altri mode switch poiché lo scheduling è non preemptive.
Per ogni processo 2N+2 mode switch. In totale (2N+2)P mode switch.
Se le system call non sono bloccanti non cambia nulla poiché l'I/O viene comunque eseguito in kernel mode.
Potrebbe capitare che tutti i processi siano bloccati. In questo caso, se facciamo l'assunzione (verosimile) che il ciclo di "idle" venga eseguito in kernel mode, l'analisi precedente è ancora valida.

Domanda 2

* Descrivi la tecnica denominata RAID 0. Spiega brevemente la sue caratteristiche rispetto a affidabilità, lettura e scrittura sequenziali di grandi quantità di dati (es. file molto lunghi), grandi quantità di letture e scritture casuali (es. sistemi transazionali).

RAID 0 prevede che i dati siano suddivisi in strip sistemati consecutivamente su N dischi.
Lettura e scrittura bulk sono estremamente più veloci perché si riesce a leggere o scrivere N strip contigui contemporaneamente.
Lettura e scrittura casuale sono estremamente più veloci poiché se gli accessi sono distribuiti uniformemente sugli N dischi riesco a servire N richieste contemporaneamente.
L'affidabilita e molto peggiore poiché basta che uno dei dischi si danneggi perché tutto il sistema sia non funzionante.



(a) RAID 0 (non-redundant)

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

Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 18 luglio 2005

Domanda 3

1. * Un processo effettua i seguenti riferimenti (in sola lettura) alle sue pagine di memoria: 6 5 4 3 4 1 5 3 6. Supponi che il sistema allochi per il processo 4 frame inizialmente vuoti. Il resident set viene gestito con una politica di page replacement LRU. Compila la seguente tabella mostrando il resident set dopo ciascun riferimento a memoria ed evidenziando i page faults.

pagina acceduta	***6***		***5***		***4***		***3***		***4***		***1***		***5***		***3***		***6***	
resident set		6		5		4		3		4		1		5		3		6
(l'ordine in cui disponi le pagine nelle caselle è irrilevante, scegli quello che più ti fa comodo)				6		5		4		3		4		1		5		3
						6		5		5		3		4		1		5
								6		6		5		3		4		1
page fault?	si		si		si		si		no		si		no		no		si	

2. ** Supponi ora che il sistema operativo abbia assegato a tale processo 2 frames gestiti con una politica FIFO e che supporti due frames di page buffering anch'essi gestiti con una politica FIFO. Supponi che nessun altro processo sia attivo sul sistema. Compila la seguente tabella con lo stato dei frame dopo ciascun riferimento a memoria, evidenziando page faults e accessi a disco.

pagina acceduta	***6***		***5***		***4***		***3***		***4***		***1***		***5***		***3***		***6***	
resident set (coda fifo)		6		5		4		3		3		1		5		3		6
				6		5		4		4		3		1		5		3
page buffer (coda fifo)						6		5		5		4		3		1		5
								6		6		5		4		4		1
accesso al disco?	si		si		si		si		no		si		no		no		si	
page fault?	si		si		si		si		no		si		si		si		si	

Domanda 4

1. * Descrivi il concetto di tabella delle pagine a più livelli e mostra i dati contenuti nelle singole righe delle tabelle.

Vedi materiale didattico e libro.

2. ** Considera una tabella delle pagine a due livelli e root page table sempre in memoria. Mostra gli indirizzi di memoria virtuale di due byte consecutivi la cui lettura può, nel caso peggiore, provocare 4 page faults. Supponi che l'architettura preveda indirizzamento virtuale a 32 bit,

ciascuna pagina sia lunga 4 Kbytes e ogni entry delle page tables, sia lunga 4 bytes. Giustificare la risposta.

In ogni pagina ci sono 1024 page table entries (PTE), indirizzabili con 10 bit e quindi ciascun indirizzo è composto da 10 bit che identificano la page table (PT), 10 bit che identificano la PTE, e 12 bit di offset.

Gli indirizzi sono x1=003FFFFF e x2=00400000. Infatti x1 è nella pagina puntata dalla PTE 3FF (l'ultima) della PT 0 e x2 è nella pagina puntata dalla PTE 0 (la prima) della PT 1. I fault possono avvenire sulla lettura di x1, di x2, di PT 0 e di PT1.
Un'altra soluzione possibile è ad esempio x1=f17FFFFF e x2=f1800000.

Domanda 5

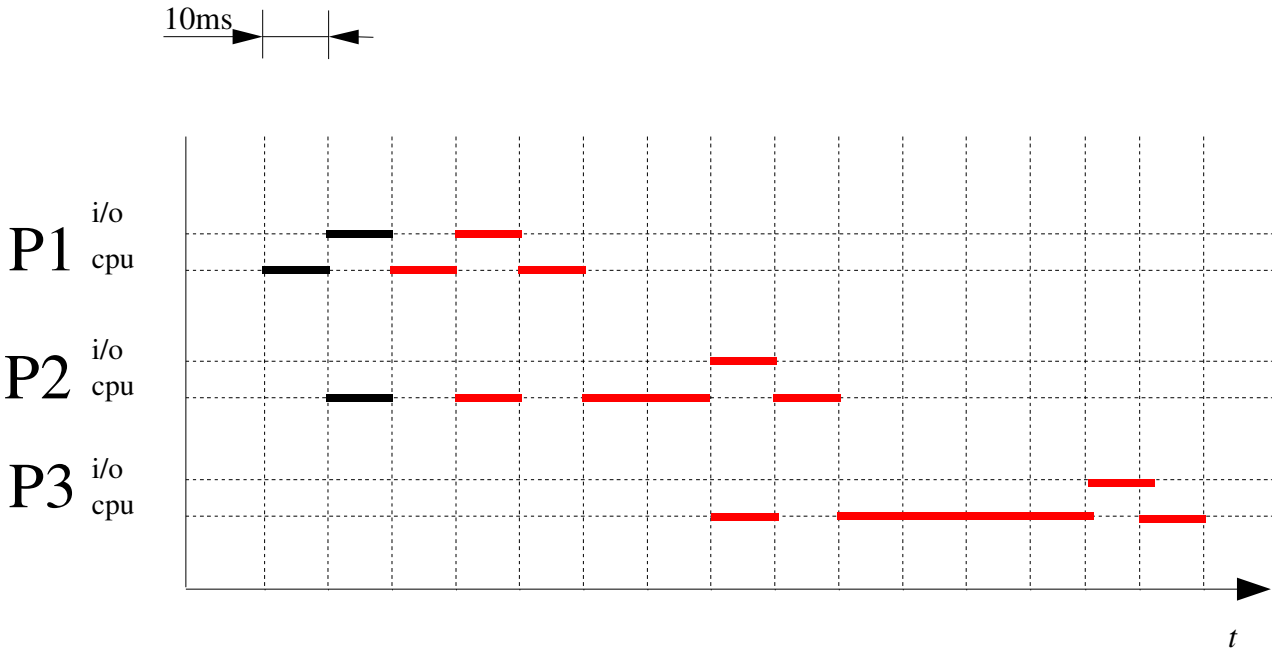
*** Considera una politica di CPU scheduling del tipo *shortest remaining time* che può fare preeption ogni volta che un processo diventa pronto per l'esecuzione e che preferisce il processo che ha il minor tempo di cpu fino al prossimo I/O. Supponi che lo scheduler conosca tale tempo. I processi P1, P2 e P3 sono caratterizzati dalle seguenti sequenze di CPU busrt e I/O burst.

P1: cpu 10ms, i/o 10ms, cpu 10ms, i/o 10ms, cpu 10ms.

P2: cpu 40ms, i/o 10ms, cpu 10ms.

P3: cpu 50ms, i/o 10ms, cpu 10ms.

Supponi che i tre processi facciano I/O su dispositivi distinti. Mostra, in ciascun istante di tempo, quali processi sono in stato running e quali in blocco, marcando, nel diagramma sottostante, le linee di cpu o di I/O corrispondenti a ciascun processo.



Cognome:_____ Nome: _____ Matricola:_____

Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 18 luglio 2005

A

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

Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 18 luglio 2005

Compito B

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

Domanda 1

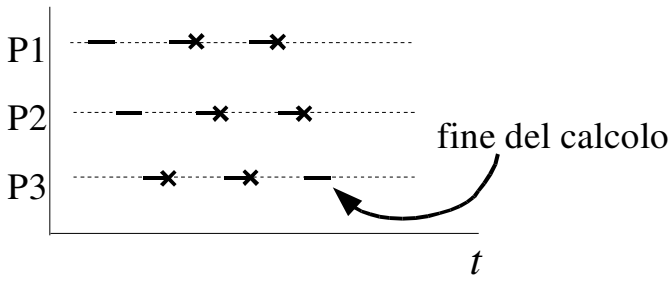
1. * Descrivi le funzionalità del dispatcher? In quali occasioni viene eseguito?

Il dispacher è quella parte del sistema operativo che ripristina lo stato del processo che deve essere eseguito di li a poco (il processo viene deciso dallo scheduler). Esso viene eseguito quando il processo che era in esecuzione deve rilasciare la CPU e cioè nelle seguenti situazioni: ha eseguito una system call bloccante, è scaduto il suo quanto di tempo, un processo a priorità maggiore diviene pronto (per scheduling preemptive),.il processo termina la sua esecuzione.

2. ** Considera un sistema con architettura "execution within user process" (come in UNIX) con scheduling non preemptive. Su tale sistema vi è un processo cpu bound, di durata infinita, che non fa alcuna system call e *P* processi I/O bound che fanno, ciascuno, *N* system call di I/O bloccanti e una system call che termina il processo. Quante volte viene invocato il dispatcher fino al momento in cui l'ultimo dei *P* processi termina? Giustificare la risposta.

Il testo di questa domanda non è corretto. Se il processo cpu bound è running allora gli altri processi vanno in starvation a causa della politica non preemptive. Vengono date delle soluzioni possibili secondo varie ipotesi di modifica della domanda.

- Supponiamo che gli I/O burst siano sempre molto brevi, e che il processo cpu bound non sia mai schedulato. Il numero delle invocazioni al dispatcher è dato dal numero di cpu bust dei processi I/O bound: (N+1)P
- Supponiamo che non ci sia il processo cpu bound, gli I/O burst non siano necessariamente brevi e che quando tutti i processi I/O bound sono bloccati vi sia un ciclo di "idle" che supponiamo non provocare l'esecuzione del dispatcher. Il numero delle invocazioni al dispatcher è dato dal numero di cpu bust dei processi I/O bound: (N+1)P
- Supponiamo che il processo cpu bound sia a priorità inferiore rispetto ai processi I/O bound e che quando questo processo è running, la politica è preemptive. In altre parole, quando un processo I/O bound diviene pronto se è in esecuzione il processo cpu bound esso viene interrotto. Il dispatcher viene invocato un numero di volte pari a (N+1)P +X dove X è il numero di volte che viene schedulato il processo cpu bound quando tutti i processi I/O bound sono bloccati. Per X si ha 0≤X≤ NP poiché nel caso peggiore il processo cpu bound parte dopo ciascuna system call di processo I/O bound tranne per le prime P-1 schedulazioni in cui sicuramente c'è almeno un processo I/O bound pronto. Ecco un esempio per N=2 e P=3, con la X vengono indicati di momenti in cui è possibile che venga schedulato il processo cpu bound.



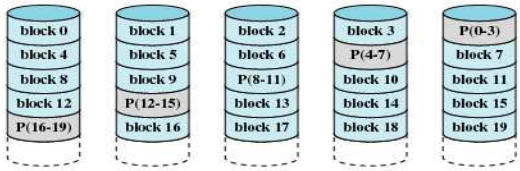
Cognome:_____ **Nome:** _____ **Matricola:**_____

Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 18 luglio 2005

Domanda 2

* Descrivi la tecnica denominata RAID 5. Spiega brevemente la sue caratteristiche rispetto a affidabilità, lettura e scrittura sequenziali di grandi quantità di dati (es. file molto lunghi), grandi quantità di letture e scritture casuali (es. sistemi transazionali).

RAID 5 prevede che i dati siano suddivisi in blocchi sistemati consecutivamente su N+1 dischi. Inoltre un blocco di parità è inserito per ciascuna "riga" come in figura.
Lettura bulk e transazionale sono estremamente più veloci (bulk xN, transazionale x(N+1)). La scrittura risente del fatto che il blocco di parità va aggiornato e quindi questo è un collo di bottiglia. L'affidabilita è buona, si può sopportare il guasto di un disco senza perdere dati.



(f) RAID 5 (block-level distributed parity)

Domanda 3

3. * Un processo effettua i seguenti riferimenti (in sola lettura) alle sue pagine di memoria: 1 6 2 3 2 4 6 3 1. Supponi che il sistema allochi per il processo 4 frame inizialmente vuoti. Il resident set viene gestito con una politica di page replacement LRU. Compila la seguente tabella mostrando il resident set dopo ciascun riferimento a memoria ed evidenziando i page faults.

pagina acceduta	***1***		***6***		***2***		***3***		***2***		***4***		***6***		***3***		***1***	
resident set		1		6		2		3		2		4		6		3		1
(l'ordine in cui disponi le pagine nelle caselle è irrilevante, scegli quello che più ti fa comodo)				1		6		2		3		2		4		6		3
						1		6		6		3		2		4		6
								1		1		6		3		2		4
page fault?	si		si		si		si		no		si		no		no		si	

4. ** Supponi ora che il sistema operativo abbia assegato a tale processo 2 frames gestiti con una politica FIFO e che supporti due frames di page buffering anch'essi gestiti con una politica FIFO. Supponi che nessun altro processo sia attivo sul sistema. Compila la seguente tabella con lo stato dei frame dopo ciascun riferimento a memoria, evidenziando page faults e accessi a disco.

pagina acceduta	***1***		***6***		***2***		***3***		***2***		***4***		***6***		***3***		***1***	
resident set (coda fifo)		1		6		2		3		3		4		6		3		1
				1		6		2		2		3		4		6		3
page buffer (coda fifo)						1		6		6		2		3		4		6
								1		1		6		2		2		4
accesso al disco?	si		si		si		si		no		si		no		no		si	
page fault?	si		si		si		si		no		si		si		si		si	

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

Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 18 luglio 2005

Domanda 4

1. * Descrivi il metodo di allocazione *buddy system* e come si comporta rispetto ai problemi della frammentazione interna ed esterna.

Vedi materiale didattico e libro.

Limitata frammentazione esterna poiché la taglia dei frammenti che non hanno buddy è sempre maggiore o uguale del blocco più piccolo allocato nel sistema.
Frammentazione interna: si ma non si spreca più della metà della memoria allocata. Secondo l'ipotesi di richieste uniformemente distribuite se ne spreca mediamente ¼.

2. ** Considera un buddy system in cui l'aggregazione dei blocchi venga effettuata, se possibile, dopo ogni deallocazione. Mostra una sequenza di allocazioni e deallocazioni in cui questa scelta è inefficiente. Suggerisci un approccio migliore.

Le sequenze in cui ciascuna allocazioni e seguita dalla rispettiva deallocazione sono inefficienti poiché il buddy system ad ogni deallocazione unisce ciò che aveva spezzato.
Un approccio migliore è quello "lazy" in cui si va in cerca di blocchi da unire solo se, in fase di allocazione, non trovo un blocco sufficientemente grande per soddisfare la richiesta.

Domanda 5

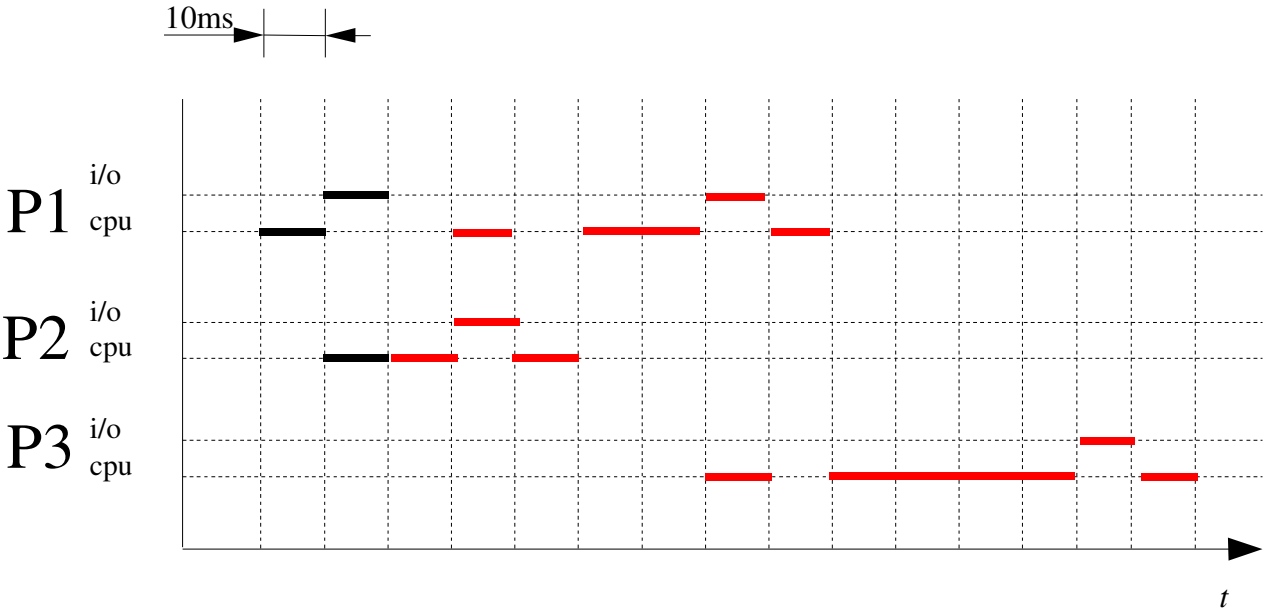
*** Considera una politica di CPU scheduling del tipo *shortest remaining time* che può fare preeption ogni volta che un processo diventa pronto per l'esecuzione e che preferisce il processo che ha il minor tempo di cpu fino al prossimo I/O. Supponi che lo scheduler conosca tale tempo. I processi P1, P2 e P3 sono caratterizzati dalle seguenti sequenze di CPU busrt e I/O burst.

P1: cpu 10ms, i/o 10ms, cpu 30ms, i/o 10ms, cpu 10ms.

P2: cpu 20ms, i/o 10ms, cpu 10ms.

P3: cpu 50ms, i/o 10ms, cpu 10ms.

Supponi che i tre processi facciano I/O su dispositivi distinti. Mostra, in ciascun istante di tempo, quali processi sono in stato running e quali in blocco, marcando, nel diagramma sottostante, le linee di cpu o di I/O corrispondenti a ciascun processo.



Cognome:_____ Nome: _____ Matricola:_____

Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 18 luglio 2005

B