SISTEMI OPERATIVI
7 febbraio 2017
corso A nuovo ordinamento
e parte di teoria del vecchio ordinamento indirizzo SR

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

1. Ricordate che non potete usare calcolatrici o materiale didattico, e che potete consegnare al massimo tre prove scritte per anno accademico.
2. Gli studenti a cui sono stati riconosciuti i 3 cfu di "linguaggio C" devono rispondere solo alle domande delle parti di teoria e di laboratorio Unix, e consegnare entro 1 ora e trenta minuti.
3. Gli studenti del vecchio ordinamento, indirizzo SR, devono rispondere solo alle domande della parte di teoria, e devono consegnare entro 1 ora.

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

ESERCIZIO 1 (5 punti)

a) (2 p.) Si consideri il problema dei lettori e scrittori visto a lezione, dove i codici del generico scrittore e del generico lettore sono riportati qui di seguito.

Inserite le istruzioni mancanti necessarie per il funzionamento del sistema secondo la soluzione vista a lezione, indicando anche il semaforo e i valori di inizializzazione mancanti.

semafori e variabili condivise necessarie con relativo valore di inizializzazione:
semaphore mutex = 1;
semaphore scrivi = 1;
int numlettori = 0;

"scrittore"
{
wait(scrivi);
Esegui la scrittura del file
signal(scrivi)
}

"lettore"
{
wait(mutex);

numlettori++;

if numlettori == 1 **wait(scrivi);**

signal(mutex);

... leggi il file ...

wait(mutex);

numlettori--;

if numlettori == 0 **signal(scrivi);**

signal(mutex);

b) Si consideri l'algoritmo del punto precedente, e si assuma che tutti i lettori e tutti gli scrittori accedano al file condiviso per una quantità di tempo finita. Spiegate se è possibile, e se si per quale ragione, che un processo vada in starvation.

Un processo lettore non può mai andare in starvation. Un processo scrittore può andare in starvation se si è addormentato sul semaforo scrivi a *causa della presenza di uno o più lettori in fase di lettura* e continuano ad arrivare nuovi lettori, di modo che la variabile numlettori non scende mai a zero.

c) riportate lo pseudocodice relativo alla corretta implementazione dell'operazione di *signal*. Dite a cosa serve la system call usata nel codice della *signal*.

Si vedano i lucidi della sezione 6.5.2.

d) per quale ragione è possibile implementare le sezioni critiche contenute nel codice della Wait e della Signal usando la disabilitazione degli interrupt, ma questa soluzione non è adottabile per le sezioni critiche dei processi degli utenti?

Perché i processi utenti non garantiscono che la disabilitazione degli interrupt avvenga per un tempo limitato, e neppure che gli interrupt vengano riabilitati all'uscita di una sezione critica.

ESERCIZIO 2 (5 punti)

In un sistema che adotta la memoria paginata le pagine hanno una dimensione tale da produrre una frammentazione interna media per processo di 1 Kbyte. Il sistema adotta una paginazione a due livelli, e la tabella delle pagine esterna più grande del sistema occupa esattamente un frame. Ogni entry di una tabella delle pagine (interna o esterna) occupa 2 byte, e tutti i bit vengono usati per scrivere il numero di un frame.

a) (2 p.) Quali sono le dimensioni degli spazi di indirizzamento logico e fisico del sistema?

Il sistema adotta pagine da 1*2 = 2 Kbyte, ossia 2^{11} byte. La tabella delle pagine esterna più grossa del sistema contiene $2^{11}/2^{1}$ entry, e quindi la tabella delle pagine interna più grande del sistema ha in tutto $2^{10}*2^{10} = 2^{20}$ entry. Lo spazio logico è quindi grande $2^{20} * 2^{11} = 2^{31}$ byte e quello fisico è grande $2^{16} * 2^{11} = 2^{27}$ byte.

b) perché un sistema con le caratteristiche del punto a) dovrebbe implementare un algoritmo di rimpiazzamento delle pagine?

Per permettere ad un insieme di processi che occupano uno spazio logico complessivo maggiore di quello fisico disponibile di girare contemporaneamente.

c) A e B sono due sistemi operativi identici che girano su due macchine identiche. L'unica differenza è che A dotato di memoria virtuale e B no. Sia T_A il tempo che intercorre tra quando un programma P viene mandato in esecuzione a quando termina sul sistema A, e sia T_B il tempo che intercorre tra quando lo stesso programma P viene mandato in esecuzione a quando termina sul sistema B. È possibile che $T_A < T_B$? (motivate la vostra risposta)

Si, ad esempio se in una particolare esecuzione di P su sul sistema A, P non genera mai page fault e non usa una porzione di codice che quindi non deve essere caricata in memoria primaria, risparmiando così sul tempo di caricamento rispetto al sistema B (che invece caricherà sempre in RAM l'intera immagine di P)

d) Si considerino i seguenti interventi su un sistema dotato di memoria virtuale:

1. salvare periodicamente sull'hard disk le pagine col dirty bit a 1.
2. usare un dispositivo di memoria secondaria più veloce.
3. aumentare la quantità di memoria primaria.
4. passare dall'algoritmo di rimpiazzamento FIFO a quello della seconda chance migliorata.
5. usare una CPU più veloce.

Quale, tra questi interventi, ha l'effetto migliore sul fenomeno del thrashing del sistema, e perché?

L'intervento 3, poiché è permette di aumentare il numero medio di pagine che un processo riesce a tenere in RAM, diminuendo così la probabilità di page fault.

Quale, tra questi interventi, non ha alcun effetto sul fenomeno del thrashing del sistema, e perché?

L'intervento 5, poiché non ha alcun effetto sulla frequenza con cui si verifica il page fault, e ha un effetto quasi nullo sul tempo di gestione del page fault.

NOTA (che non era necessario includere nella risposta): l'intervento 1 aumenta solo la probabilità che, in caso di page fault, la pagina vittima non debba essere salvata prima di essere sovrascritta; l'intervento 2 diminuisce solo il tempo di gestione del page fault, l'intervento 4 può, in alcuni casi, ridurre leggermente la frequenza del page fault. In generale l'intervento più significativo per il controllo e la prevenzione del thrashing è sempre quello di limitare opportunamente il grado di multiprogrammazione del sistema (si veda ad esempio la sezione 9.6.3)

ESERCIZIO 3 (5 punti)

Un sistema operativo adotta l'allocazione indicizzata dei file, e usa un hard disk da 16 Gigabyte formattato in blocchi da 1024 byte.

a) Qual è lo spazio su disco "sprecato" per tenere traccia di tutti i blocchi di dati di un file grande 300 Kbyte? (esplicitate le assunzioni che è necessario fare per rispondere alla domanda).

Assumendo di usare 4 byte per scrivere il numero di un blocco del file, un blocco indice può tenere traccia al massimo di 256 blocchi. Assumendo di adottare una allocazione indicizzata a schema concatenato, due blocchi indice sono sufficienti a tenere traccia dell'intero file, e dunque lo spazio "sprecato" ammonta ai 2048 byte dei due blocchi indice. (si noti che si poteva anche assumere di usare solo 3 byte per scrivere il numero di un blocco, e in questo caso un solo blocco indice sarebbe stato sufficiente a tenere traccia di tutti i blocchi di dati del file, con uno spreco di 1024 byte).

b) Se invece il sistema in oggetto usasse la variante efficiente dell'allocazione concatenata, quanto sarebbe grande, in megabyte, la FAT di questo sistema? (motivate numericamente la vostra risposta esplicitando eventuali assunzioni)

La FAT è un array con una entry per ciascun blocco dell'hard disk e che contiene il numero di un blocco, per cui (assumendo di usare 3 byte per scrivere il numero di un blocco): 2^{24} x 3 byte = 48 megabyte

c) Quali sono gli svantaggi nell'uso della FAT?

Per garantire un accesso efficiente ai file deve essere sempre tenuta in RAM occupando spazio, se viene persa si perdono tutte le informazioni sul file system, e deve quindi essere periodicamente salvata su disco.

d) Se un hard disk è al 99,99% occupato da file molto piccoli (ossia di dimensione inferiore ad un blocco dell'hard disk), quale tipo di allocazione dello spazio su disco converrebbe adottare, e perché?.

Se si considerano solo le tre tecniche di base, l'allocazione contigua sarebbe sicuramente la migliore, seguita da quella concatenata (che però spreca alcuni byte di ciascun blocco per il puntatore – non usato – al blocco successivo). Se si includono anche le altre varianti viste a lezione, NTFS sarebbe una soluzione con prestazioni simili all'allocazione contigua.

e) Si considerino le tecniche di allocazione dello spazio su disco adottate da Windows NTFS e da Unix. Quale dei due sistemi offre le prestazioni migliori in termini di tempo di accesso ai dati dei file piccoli?

NTFS, in quanto memorizza i dati di un file piccolo insieme agli altri attributi nell'elemento che tiene traccia del file stesso.