SISTEMI OPERATIVI
4 luglio 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 15 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) 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 una delle soluzioni viste a lezione, indicando il semaforo (o i semafori) mancante/i e il relativo valore di inizializzazione.

semafori e variabili condivise necessarie con relativo valore di inizializzazione:
semaphore mutex = 1;
semaphore scrivi = 1; (ed eventualmente, semaphore leggi = 1 per la versione "fair" dell'algoritmo)
int numlettori = 0;

"scrittore"
{
wait (leggi); // per la versione "fair" dell'algoritmo
wait(scrivi);
Esegui la scrittura del file
signal(scrivi)
signal (leggi); // per la versione "fair" dell'algoritmo
}

"lettore"
{
wait (leggi); // per la versione "fair" dell'algoritmo
wait(mutex);

numlettori++;

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

signal(mutex);
signal (leggi); // per la versione "fair" dell'algoritmo

... leggi il file ...

wait(mutex);

numlettori--;

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

signal(mutex);

b) Se esistono, indicate un caso in cui a) un processo lettore può andare in starvation; b) un processo scrittore può andare in starvation.

Caso della soluzione "reader first": processo lettore: se il processo che sta scrivendo non termina mai; processo scrittore: se è in attesa di scrivere perché ci sono processi in lettura, e continuano ad arrivare nuovi lettori.

Caso della soluzione "fair": se il processo che sta scrivendo o leggendo non termina mai.

c) Cosa vuol dire che "un algoritmo di scheduling" soffre di starvation? Come si può risolvere questo problema?

Vuol dire che non garantisce a un processo in coda di ready di poter usufruire della CPU in un tempo limitato. Con un meccanismo di aging.

d) Riportate lo pseudocodice che descrive l'implementazione dell'operazione di wait, spiegate perché questa implementazione non fa sprecare inutilmente tempo di cpu se il processo che la chiama trova il valore della variabile semaforica a 0.

Per lo pseudocodice si vedano i lucidi della sezione 6.5.2

Perché il pcb del processo che si addormenta sul semaforo in attesa di entrare in sezione critica viene messo nella coda di waiting del semaforo e non può più essere selezionato dallo scheduler fino a che la sezione critica non si libera e il processo addormentato viene riportato in coda di ready.

ESERCIZIO 2 (5 punti)

Si consideri un sistema in cui in una tabella delle pagine di un processo l'indice più grande usabile nella tabella delle pagine di quel processo può essere 3FFF. Un indirizzo fisico del sistema è scritto su 26 bit, e la RAM è suddivisa in 8000 (esadecimale) frame.

a)
Quanto è grande, in megabyte, lo spazio di indirizzamento logico del sistema (esplicitate i calcoli che fate)?

8000(esadecimale) = 2^{15}, per cui un numero di frame è scritto su 15 bit, e la dimensione di un frame, e quindi di una pagina, è di $2^{\wedge 11}$ byte (26 - 15 = 11). Poiché il numero più grande di una pagina è 3FFF, ci possono essere al massimo $2^{\wedge 14}$ pagine, e lo spazio di indirizzamento logico è di $2^{\wedge 14}$ x $2^{\wedge 11}$ byte (pari a circa 32 megabyte).

b)
Se il sistema adotta una paginazione a due livelli, quanto spazio occupa, in byte, la tabella delle pagine *esterna* più grande di questo sistema? (motivate numericamente la vostra risposta).

La tabella delle pagine interna più grande del sistema indirizza $2^{\wedge 14}$ pagine, e ogni entry contiene il numero di un frame, per cui sono necessari almeno due byte per ogni entry. La dimensione di quella tabella sarà quindi di $2^{\wedge 14} \times 2 = 2^{\wedge 15}$ byte (ossia maggiore della dimensione di un frame, il che implica appunto la necessità – in linea teorica – di paginare la tabella delle pagine interna). $2^{\wedge 15}$ byte = 32 Kbyte, e quindi questa tabella delle pagine interna occupa 16 frame. Di conseguenza la tabella esterna sarà composta da 16 entry di due byte ciascuno, ossia 32 byte.

c)
In quale caso il sistema descritto qui sopra deve implementare la memoria virtuale?

> Lo spazio di indirizzamento logico è minore di quello fisico, per cui l'unico caso è quello in cui si vogliano avere in esecuzione contemporaneamente processi che, insieme, occupano uno spazio maggiore dello spazio di indirizzamento fisico.

d)
Un sistema (hardware + sistema operativo) soffre spesso del problema del thrashing. Indicate due modifiche al sistema, una hardware e una software, che potrebbero migliorare la situazione. Motivate le indicazioni che date.

Modifica Hardware: aggiungere più ram, cosicché ogni processo del sistema avrà mediamente più frame a disposizione e meno probabilmente genererà un page fault.
Modifica software: diminuire il grado di multiprogrammazione, in modo che meno processi hanno a disposizione ciascuno più frame.

ESERCIZIO 3 (5 punti)

Sia dato un sistema operativo Unix, in cui viene eseguito correttamente il comando:
"mkdir /users/gunetti/myfiles/pippo" (dove "pippo" non esisteva prima dell'esecuzione del comando)

a) cosa succede dal punto di vista degli hard link coinvolti nel comando?

L'hard link di "myfiles" viene incrementato di uno, e l'hard link di "pippo" all'interno del nuovo i-node associato viene creato e inizializzato al valore 2.

b) vengono ora dati i comandi:
cd /users/gunetti/myfiles
ln –s pippo pluto
ln pluto paperino

Che effetto ha il comando 3? (motivate la vostra risposta)

Il terzo comando fallisce, dato che gli hard link a directory non sono permessi.

c) Si assuma ora che nella cartella pippo sia presente il file di testo A, e si vuole creare un collegamento veloce ad A nella radice del file system. Conviene usare un hard link o un symbolic link? (motivate la vostra risposta)

Un hard link, che permette di raggiungere più velocemente i dati di A. Col symbolic link infatti, si crea un nuovo i-node che rimanda all'i-node di A.

d) occupa più spazio un hard link o un symbolic link? (motivate la vostra risposta)

Un symbolic link, perché alloca un nuovo i-node.

e) perché in un file system sono necessari i simbolic link (motivate la vostra risposta)?

Perché permettono di costruire collegamenti tra directory ma (essendo riconoscibili dai comandi e dagli applicativi che possono visitare ricorsivamente porzioni del file system) permettono di evitare cicli all'interno del file system (in sostanza, perché permettono di costruire file system che hanno la struttura di un grafo aciclico)