SISTEMI OPERATIVI
28 gennaio 2019
corso A nuovo ordinamento

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

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

ESERCIZIO 1 (7 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 la soluzione vista 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;
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) Questo algoritmo soffre di starvation? (motivate la vostra risposta)

Si per un processo scrittore: se è in attesa di scrivere perché ci sono processi in lettura, e continuano ad arrivare nuovi lettori.

c) Riportate lo pseudocodice che descrive la corretta implementazione delle operazioni di wait e di signal, spiegate perché questa implementazione è migliore di una soluzione basata sul busy-waiting

Per lo pseudocodice si vedano i lucidi della sezione 6.5.2

Perché non fa sprecare inutilmente tempo di CPU ai processi che si addormentano sul semaforo

d) In quali modi un sistema operativo mantiene sempre un certo controllo della macchina anche mentre non sta girando codice del sistema operativo stesso?

Uso di istruzioni privilegiate per eseguire operazioni delicate
Uso di un timer hardware
Uso di registri appositi per il controllo degli indirizzi usati nelle istruzioni dei programmi utente

e) Riportate il diagramma di stato della vita di un processo.

si vedano i lucidi della sezione 3.1.

ESERCIZIO 2 (7 punti)

Di un sistema è noto che:
la tabella delle pagine più grande del sistema occupa esattamente un frame
il numero di un frame è scritto su 2 byte usando tutti i bit a disposizione
la frammentazione interna media prodotta da un processo del sistema è di circa 4 Kbyte

a) quanto sono grandi lo spazio di indirizzamento fisico e logico del sistema? (motivate numericamente la vostra risposta)?

Poiché la frammentazione interna media di un sistema paginato è di circa mezza pagina, le pagine del sistema sono grandi 8192 byte, ossia 2^{13} byte, e la PT più grande del sistema ha $2^{13}/2 = 2^{12}$ entry. Lo spazio logico è quindi di 2^{25} byte e lo spazio fisico è di $2^{\wedge 29}$ byte.

b) In quale caso il sistema sopra descritto dovrebbe implementare la memoria virtuale?

Poiché lo spazio logico è inferiore a quello fisico, la MV è necessaria solo se vogliamo far girare contemporaneamente un insieme di processi che occupano uno spazio maggiore di quello fisico disponibile.

c) cosa significa che un processo è nello stato “waiting for page”?

che mentre era in esecuzione ha indirizzato una pagina non presente in ram, ed è stato messo nello stato indicato nell’attesa che il SO recuperi e trasferisca in ram la pagina mancante.

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.

e) indicate gli **svantaggi** della paginazione della memoria

si vedano i lucidi della sezione 8.4.1

f) cosa significa che un sistema adotta un *binding dinamico degli indirizzi*?

Che un programma che gira su quel sistema usa solo indirizzi relativi, e la traduzione degli indirizzi da relativi ad assoluti avviene durante l'esecuzione delle istruzioni che usano quegli indirizzi

ESERCIZIO 3 (6 punti)

Un hard disk ha la capacità di 128 gigabyte, è formattato in blocchi da 800 (esadecimale) byte, e usa una qualche forma di allocazione indicizzata per memorizzare i file su disco. Sull'hard disk è memorizzato un file A grande 1,5 Megabyte. Nel rispondere alle domande sottostanti, specificate sempre le assunzioni che fate.

a) Quante operazioni di I/O su disco sono necessarie per portare in RAM l'ultimo blocco del file A, assumendo che inizialmente sia presente in RAM solo la copia del file directory che "contiene" il file?

L'hard disk contiene $2^{37}/2^{11} = 2^{26}$ blocchi, e sono quindi necessari 4 byte per scrivere in numero di un blocco. Un blocco indice può quindi contenere al massimo 2048/4 = 512 numeri/puntatori a blocco. La risposta dipende poi dal tipo di allocazione indicizzata assunta:

1) Allocazione indicizzata a schema concatenato: sono necessari 2 blocchi indice per tenere traccia di tutti i blocchi del file. Se assumiamo che sia già in RAM il numero del primo blocco indice, sono necessarie 3 operazioni di I/O: lettura dei due blocchi indice più lettura del blocco del file.
2) Allocazione indicizzata a più livelli. È sufficiente usare uno schema a due livelli. Se assumiamo che sia già in RAM il numero del blocco indice esterno, sono necessarie 3 operazioni di I/O: lettura del blocco indice esterno, lettura di un blocco indice interno, lettura del blocco del file.
3) Allocazione indicizzata Unix: è necessario usare il puntatore a di doppia in direzione, quindi, assumendo già in RAM il numero dell'index-node, sono necessarie 4 operazioni di I/O: lettura dell'index-node, lettura del blocco indice puntato dal puntatore di doppia in direzione, lettura del blocco indice interno, lettura del blocco del file.

b) 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)
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.

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) In quale caso l'accesso in lettura ad un file memorizzato su un sistema RAID non è più veloce che se il file fosse memorizzato su un normale hard disk?

Quando il file è memorizzato su uno più blocchi appartenenti a strip contenuti sullo stesso disco del RAID (e il RAID usato non è di tipo 1, poiché in questo caso, se il file è memorizzato su almeno due strip, si può sfruttare il disco di mirroring)