SISTEMI OPERATIVI
28 gennaio 2020 - corso A

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

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

ESERCIZIO 1 (7 punti)

a) Riportate lo pseudocodice che descrive l'implementazione delle operazioni di wait e di signal, spiegate perché questa implementazione non fa sprecare inutilmente tempo di cpu se il processo che chiama la wait 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 dall'invocazione della signal su quel semaforo da parte di un altro processo.

b1) Riportate, a vostra scelta, il *diagramma di transizione degli stati di un processo* o il *diagramma di accodamento* di un sistema time sharing **che usa un algoritmo di rimpiazzamento delle pagine**;

b2) Facendo riferimento al sistema del punto b1), elencate le ragioni per cui un processo può passare da un qualsiasi stato *waiting* allo stato *ready to run*

b1) Per i diagrammi si vedano le slide della sezione 3.1.2 e 3.2.1 e il lucido a p. 9 del cap. 9.

b2) è stata completata una operazione di I/O; signal sul semaforo su cui il processo era addormentato; è arrivata in ram la pagina mancante che aveva causato un page fault del processo.

c) In quali modi il sistema operativo conserva sempre il controllo della macchina e si protegge da eventuali malfunzionamenti dei programmi degli utenti?

Uso di istruzioni privilegiate eseguibili solo dal codice del SO, uso di un timer hardware che restituisce il controllo al SO allo scadere del tempo stabilito, protezione della memoria (uso di registri base e limite o paginazione della memoria)

d) Che svantaggio da l'uso dei processi al posto dei thread?

Un insieme di peer thread condivide lo spazio di indirizzamento. Per questa ragione, il context switch tra peer thread e la creazione di un nuovo peer thread richiedono molto meno tempo delle corrispondenti operazioni sui processi.

e) Tre processi P1 P2 e P3 arrivano uno dopo l'altro – nell'ordine indicato ma in tempi diversi – in coda di ready. I tre processi sono caratterizzati da un unico burst di CPU e da nessun burst di I/O. In quale caso

il waiting time medio del sistema relativo ai tre processi risulta lo stesso sia adottando un algoritmo di scheduling FCFS che adottando una algoritmo di scheduling SJF preemtpive?

È sufficiente che: Burst_P1 ≤ Burst_P2 ≤ Burst_P3

ESERCIZIO 2 (7 punti)

In un sistema operativo un indirizzo fisico è scritto su 40 bit, e l'offset più grande in un frame è 7FF. Si sa che la tabella delle pagine più grande del sistema occupa esattamente 400_{16} (400 esadecimale) frame.

a) Qual è lo spazio di indirizzamento logico del sistema? (esplicitate i calcoli che fate)

L'offset all'interno di un frame è scritto su 11 bit, dunque nel sistema ci sono 40-11 = 2^{29} frame, e per scrivere il numero di un frame all'interno di una entry della tabella delle pagine sono necessari 29 bit, che arrotondiamo a 4 byte. Dunque, la tabella delle pagine più grande del sistema avrà $2^{10}*2^{11}/2^2 = 2^{19}$ entry, corrispondenti ad una suddivisione dello spazio logico in 2^{19} pagine. La dimensione dello spazio di indirizzamento logico è quindi di $2^{19}*2^{11} = 2^{30}$ byte = 1 gigabyte.

b) Il sistema potrebbe soffrire di thrashing? (motivate la risposta)

Solo se si vuole implementare la memoria virtuale (cosa non necessaria dato che lo spazio di indirizzamento fisico è maggiore di quello logico)

c) Come è fatta una Inverted Page Table?

Contiene un numero di entry pari al numero di frame in cui è suddivisa la memoria fisica, e ogni entry è formata da una coppia: numero di una pagina, PID del processo a cui quella pagina appartiene.

d) Perché le librerie dinamiche sono da preferire alle librerie statiche?

Perché possono essere condivise tra più processi, per cui occupano meno spazio in RAM di quelle statiche. Vengono caricate in RAM solo se sono chiamate, per cui i processi partono più rapidamente. Se vengono aggiornate non occorre ricompilare i programmi che le usano.

e) E' vera la seguente affermazione? (motivate la vostra risposta) *In un qualsiasi sistema operativo che implementi la memoria virtuale, l'effettivo tempo di esecuzione di un programma dipende molto anche dal modo con cui accede ai propri dati in RAM.*

L'affermazione è vera. Si veda ad esempio la sezione 10.9.5 (struttura dei programmi).

f) Se un sistema soffre spesso di thrashing, qual è l'intervento hardware o software che molto probabilmente avrà l'impatto migliore sulle prestazioni del sistema, e perché?

Aumentare la quantità di RAM, poiché in questo modo i processi avranno più frame a disposizione e meno probabilmente causeranno page fault. In alternativa, limitare il grado di multiprogrammazione.

ESERCIZIO 3 (6 punti)

a) Un hard disk ha la capienza di 2^{40} byte, ed è formattato in blocchi da 1024 byte. Quanti accessi al disco sono necessari per leggere l'ultimo blocco di un file A della dimensione di 2^{15} byte, assumendo che sia già in RAM il numero del primo blocco del file stesso e che venga adottata una allocazione concatenata dello spazio su disco? (motivate la vostra risposta)

33. Ogni blocco infatti memorizza 1020 byte di dati più 4 byte di puntatore al blocco successivo (infatti, $2^{40}/2^{10} = 2^{30}$), per cui sono necessari 33 blocchi per memorizzare l'intero file (possiamo alternativamente osservare che se potessimo usare un intero blocco per i dati del file, ci vorrebbero $2^{15}/2/^{10} = 2^5 = 32$ blocchi per memorizzare il file, ma poiché in ognuno dei 32 blocchi 4 byte sono usati per contenere il puntatore al blocco successivo, i blocchi effettivi in cui viene memorizzato il file sono 33)

b) Se sull'hard disk si adottassero cluster da 4 blocchi, quanto sarebbe grande, in megabyte o gigabyte, la FAT di questo sistema? (motivate numericamente la vostra risposta)

La FAT è un array con una entry per ciascun blocco dell'hard disk e che contiene il numero di un blocco. Poiché però lavoriamo con cluster da 4096 = 2^{12} byte, la FAT avrà $2^{40}/2^{12}$ entry per cui: 2^{28} x 2^2 byte = 1 gigabyte

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

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

d) Disegnate la FAT di un hard disk formato da 2^4 blocchi e contenente un unico file memorizzato, nell'ordine, nei blocchi 9, 5, 11, 3. Indicate anche dove è memorizzato il numero del primo blocco del file.

Si veda la figura 14.6 della sezione 141.4.2 delle slide.

e) Usando sempre i dati del punto a), se invece si usasse la tecnica di allocazione dello spazio su disco usata da Unix, quanti accessi al disco sarebbero necessari per leggere l'ultimo blocco di dati del file A grande 2^{15} byte, assumendo che sia già in RAM il numero dell'index-node associato all'hard link A? (motivate la vostra risposta)

3. Nell'allocazione indicizzata, il file occupa 32 blocchi dell'hard disk. Per tenere traccia di tutti i blocchi di dati di A dobbiamo usare il puntatore di singola in direzione, che punta ad un blocco indice che contiene i numeri dei blocchi di dati del file successivi ai primi 10. Dunque sarà necessario: leggere sull'hard disk l'index-node di A, leggere il blocco indice puntato dal puntatore di singola in direzione, leggere il blocco di dati.

f) qual è la differenza tra un pathname assoluto e uno relativo?

Il pathname assoluto inizia sempre con il carattere / o \ ed è specificato a partire dalla radice del file system. Unpathname assoluto non inizia mail con \ o / ed è specificato a partire da una qualsiasi directory diversa dalla radice.