

ESERCIZI/DOMANDE AGGIUNTIVE SUI PROCESSI

1) Elencate i 3 problemi che possono sorgere da un uso errato dei semafori

violazione della mutua esclusione, deadlock, starvation.

2) Cosa è una *sezione critica*?

Una porzione di codice in cui vengono usate variabili condivise con altri processi, e che deve essere eseguita senza intrecciarsi con l'esecuzione del codice degli altri processi in cui vengono accedute le stesse variabili condivise.

3) Perché una soluzione corretta al problema della sezione critica non deve dipendere dallo scheduling della cpu, ossia dall'ordine con cui i processi vengono eseguiti?

perché quell'ordine non è noto e può cambiare da esecuzione a esecuzione.

4) Definite i concetti di deadlock e starvation. Quale di questi due fenomeni è più grave, secondo voi, e perché?

Deadlock: due o più processi fermi in una attesa circolare. Starvation: un processo che non riesce a portare avanti la propria computazione. Il deadlock è più grave, perché coinvolge più processi. Inoltre, deadlock ==> starvation, ma non viceversa.

5) Qual è la differenza tra un **algoritmo di scheduling con diritto di prelazione** e un **Kernel con diritto di prelazione**?

Nel secondo, è l'esecuzione di codice del kernel che può essere interrotta.

6) Un sistema operativo usa un algoritmo di scheduling Round Robin e lascia che i processi utente si sincronizzino fra loro usando un meccanismo di sincronizzazione via hardware (ad esempio implementato attraverso l'istruzione macchina Test&Set) basato su busy waiting. Cosa succede quando un processo cerca di entrare in una sezione critica già occupata?

spreca tutto il suo quanto di tempo.

7) perché **SJF non preemptive** può produrre starvation?

Perché mentre un processo P è in RQ potrebbero arrivare continuamente in RQ altri processi con un burst di CPU inferiore a quello di P

8) se un algoritmo di scheduling a priorità è soggetto a starvation, come lo si può evitare?

con un meccanismo di aging.

9) in quali casi un processo in coda di ready può decidere di passare allo stato running?

mai. E' il SO a spostare i processi da uno stato all'altro.

10) che vantaggio da l'uso dei thread al posto dei processi?

creazione e context switch sono molto più efficienti. Condivisione automatica di variabili e strutture dati.

11) All'interno di un sistema operativo che implementa la memoria virtuale, un certo processo P è correntemente in stato di "*Waiting for page*", e si sa che, una volta ritornato in stato "*running*" non dovrà più rilasciare la CPU volontariamente prima di aver terminato la propria esecuzione (in altre parole, non deve più eseguire operazioni di I/O, di sincronizzazione o di comunicazione con altri processi, e non genererà più alcun page fault).

Quale/quali, tra gli algoritmi di scheduling **FCFS**, **SJF preemptive**, **SJF non-preemptive**, **Round Robin** garantisce/garantiscono che il processo P riuscirà a portare a termine la propria computazione? (motivate la vostra risposta, assumendo che SJF possa effettivamente essere implementato)

FCFS e RR, perché SJF soffre di starvation.

ESERCIZI AGGIUNTIVI SULLA MEMORIA PRINCIPALE E VIRTUALE

1) Un costruttore dichiara che il tempo di accesso a una locazione in memoria principale di un suo computer di recente progettazione è di circa 100 nanosec. In quali casi il tempo di accesso effettivo medio osservato durante il funzionamento della macchina può discostarsi da quanto dichiarato nelle specifiche?

Quando il SO usa una memoria paginata, eventualmente con memoria virtuale

2) Descrivete una approssimazione dell'algoritmo LRU comunemente usata, e dite se questa variante soffre dell'anomalia di Belady.

Qualsiasi variante di LRU che possa ridursi ad un algoritmo FIFO (come seconda chance e seconda chance migliorato) soffre dell'anomalia di Belady.

3) perché i sistemi operativi moderni non usano l'allocazione contigua dello spazio in RAM a partizioni fisse o a partizioni variabili?

Non usano l'allocazione a partizioni fisse perché limita a priori il grado di multiprogrammazione e soffre in modo eccessivo del problema della frammentazione interna. Non usano l'allocazione a partizioni variabili perché soffre del problema della frammentazione esterna e costringe periodicamente al ricompattamento dello spazio in RAM.

4) 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

5) Perché si usa una inverted page table?

Nei sistemi con spazi di indirizzamento logici molto grandi, per evitare di dover ricorrere ad una paginazione a molti livelli, che potrebbe rendere la traduzione degli indirizzi da logici a fisici molto onerosa quando non è possibile ricorrere al TLB

6) Descrivete il fenomeno del thrashing e indicate due modi in cui lo si può evitare.

un fenomeno che può verificarsi nei sistemi che implementano la Memoria Virtuale, in cui i processi generano continuamente page fault e passano la maggior parte del loro tempo nello stato "waiting for page" anziché portare avanti la loro computazione. Si può evitare/limitare aumentando la quantità di RAM disponibile e/o diminuendo il grado di multiprogrammazione, e implementando una qualche forma di monitoraggio che mantenga il numero di page fault in un intervallo accettabile

7A) Un sistema paginato ha una memoria fisica di 8k byte suddivisa in 16 frame. La page table di un processo ha la dimensione massima di un frame, e in questo caso contiene 0x100 (esadecimale) entry.

Quali sono, in esadecimale, l'offset massimo possibile in un indirizzo di questo sistema, e la dimensione dello spazio di indirizzamento logico?

Offset massimo = 1FF. Infatti ogni frame è grande $8k/16 = 512$ byte, e i byte vanno dallo 0-esimo al 511-esimo. Spazio Logico = 2^{17} byte. (Infatti un processo può essere formato al massimo da 0x100 esadecimale = 2^8 entry, e una pagina è grande 2^9 byte

7B) Se nel sistema considerato sono mediamente attivi 3 processi, qual è la quantità di memoria massima che viene sprecata a causa del meccanismo di allocazione dei processi in RAM (trascurando lo spazio necessario per memorizzare le page table dei processi)? Motivate la vostra risposta.

Nel caso peggiore, per ogni processo l'ultimo frame viene occupato per un solo byte, e dunque il resto del frame viene sprecato, per un totale di spreco di $3 * 511$ byte.

8A) Un sistema ha una memoria RAM formata da 256^2 frames di 256 byte ciascuno. Lo spazio di indirizzamento logico è grande $0x100$ (in esadecimale) volte quello fisico.

quali sono l'indirizzo fisico e logico più grandi ammessi dal sistema?

La RAM è formata da $2^{16} * 2^8 = 2^{24}$ byte, il cui indirizzo va da 0 a $2^{24} - 1$, pari a FF FFFF

Lo spazio logico è pari a $2^8 * 2^{24} = 2^{32}$, e l'indirizzo dell'ultimo byte è quindi FFFF FFFF

8B) Le tabelle delle pagine di questo sistema, potrebbero dover essere a loro volta paginate (motivate la vostra risposta)?

Una page table può dover essere paginata se non può essere contenuta interamente in un unico frame. Il numero di un frame è scritto su 16 bit (2 byte), per cui un frame può contenere al massimo una page table con 128 entry. Poiché una page table del sistema può essere molto molto più grande, può dover essere necessario usare una paginazione a più livelli.

9A) In un sistema paginato, l'offset all'interno di una pagina è scritto su 2 byte (usando tutti i bit disponibili), e la RAM è suddivisa in 128 frame.

Qual è l'indirizzo (in esadecimale) dell'ultimo byte di memoria della RAM?

spazio fisico = $2^{16} * 2^7$ byte, dunque un indirizzo fisico è scritto su 23 bit, e l'indirizzo dell'ultimo byte di memoria fisica sarà: 7F FFFF

9B) E' possibile sapere su quanti bit è scritto un indirizzo logico? Se si indicasse il numero di bit, se no dite perché non è possibile.

Non è possibile, poiché non è noto il numero di pagine dello spazio di indirizzamento logico.

9C) A vostro giudizio, nel sistema descritto, il problema della frammentazione è molto accentuato o incide poco su un uso efficiente della RAM? (giustificate la vostra risposta)

Le pagine/frame hanno una dimensione molto grande ($2^{16}=64k$ byte), per cui è presumibile un livello di frammentazione piuttosto alto.

9D) Nel sistema sopra descritto, si sa che, in media, sono presenti contemporaneamente 5 processi che occupano ciascuno 10 frame. Qual è quindi la quantità di RAM mediamente sprecata a causa della frammentazione? (è sufficiente riportare solo la formula da usare per il calcolo)

5*64k/2 byte

10A) Un sistema che implementa la memoria virtuale e che adotta una politica di sostituzione delle pagine locale può soffrire di trashing? (motivate la risposta)

Si. Se ci sono troppo processi nel sistema, ciascuno avrà pochi frame a disposizione e produrrà continuamente dei page fault, sostituendo una propria pagina che sarà molto probabilmente riferita nell'immediato futuro.

10B) Un sistema che implementa la memoria virtuale e che ha uno spazio di indirizzamento fisico maggiore di quello logico può soffrire di trashing? (motivate la risposta)

Si. Se un sistema implementa la memoria virtuale può soffrire di trashing indipendentemente dalla dimensione degli spazi di indirizzamento logici e fisici.

ESERCIZI AGGIUNTIVI SULLA MEMORIA SECONDARIA

1) Considerate la seguente sequenza di comandi Unix (assumete che tutti i comandi lanciati possano essere correttamente eseguiti):

```
1: cd /tmp
2: mkdir mynewdir
3: cd mynewdir
4: echo "ciao" > pippo          // crea un nuovo file di nome pippo contenente la stringa ciao
5: ln pippo pluto
6: ln -s pippo paperino
7: ln pluto topolino
8: rm pippo
9: cat topolino                 // cat stampa il contenuto del file passato come argomento
10: cat paperino
```

Dopo l'esecuzione di tutti i comandi:

qual è il valore del link counter nell'index-node associato al link fisico topolino?
qual è il valore del link counter nell'index-node associato al link fisico mynewdir?
cosa possiamo dire del link counter dell'index-node associato al link fisico tmp?
Qual è l'output del comando numero 10?

Dopo l'esecuzione di tutti i comandi:

qual è il valore del link counter nell'index-node associato al link fisico topolino? **2**
qual è il valore del link counter nell'index-node associato al link fisico mynewdir? **2**
cosa possiamo dire del link counter dell'index-node associato al link fisico tmp? **Che è aumentato di 1, a causa dell'entry ".." inserita dentro la nuova sottocartella mynewdir.**
Qual è l'output del comando numero 10? **"no such file or directory"**

2) In una cartella A di un sistema Unix viene creato un nuovo file di testo di nome *pippo*. Assumendo che la working directory sia A, indicate i comandi necessari per eseguire le seguenti operazioni (assumendo che ogni comando tenga conto di quelli eseguiti prima di lui)

- 1) creare un nuovo hard link di nome *pluto* a *pippo*:
- 2) creare un nuovo hard link di nome *topolino* a *pluto*:
- 3) creare un nuovo symbolic link di nome *minnie* a *topolino*:
- 4) copiare *pluto* in *paperino* (*paperino* prima non esisteva):
- 5) rimuovere *pippo*:
- 6) qual è il valore del link counter di *topolino* dopo il comando 3)?
- 7) qual è il valore del link counter di *pluto* dopo il comando 5)?

- 1) creare un nuovo hard link di nome *pluto* a *pippo*: **ln pippo pluto**
- 2) creare un nuovo hard link di nome *topolino* a *pluto*: **ln pluto topolino**
- 3) creare un nuovo symbolic link di nome *minnie* a *topolino*: **ln -s topolino minnie**
- 4) copiare *pluto* in *paperino* (*paperino* prima non esisteva): **cp pluto paperino**
- 5) rimuovere *pippo*: **rm pippo**
- 6) qual è il valore del link counter di *topolino* dopo il comando 3)? **3**
- 7) qual è il valore del link counter di *pluto* dopo il comando 5)? **2**

3) che differenza c'è tra un sistema RAID configurato al livello 4 e un sistema RAID configurato al livello 5? Quale delle due configurazioni è preferibile?

Nel livello 5 gli strip di parità sono distribuiti omogeneamente tra tutti i dischi, che quindi vengono sollecitati in modo uniforme. Nel livello 4 gli strip di parità risiedono su un unico disco, che viene quindi sollecitato mediamente più di tutti gli altri, dovendo essere aggiornato ad ogni modifica, aggiunta o cancellazione di un qualsiasi file

4A) 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 index-node coinvolti nel comando?

il link counter dell'index-node associato all'hard link "myfiles" viene incrementato di 1. Il link counter dell'index-node appena allocato e associato all'hard link "pippo" viene inizializzato a 2.

4B) vengono ora dati i comandi:

```
cd /users/gunetti/myfiles  
ln pippo pluto
```

cosa succede? (motivate la vostra risposta)

viene generato un messaggio di errore, perché si sta tentando di costruire un hard link ad una cartella, cosa proibita per evitare la comparsa di cicli all'interno del file system

4C) 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)

conviene usare un hard link, perché permetterà di raggiungere A senza aprire un index-node in più (quello che sarebbe associato ad un eventuale link simbolico ad A)

5A) Una cartella Unix DIR ha un link counter pari a 5. Che informazioni ci da questo valore, e come viene ricavata questa informazione?

Che nella cartella DIR sono state create tre sottocartelle. L'informazione può essere ricavata sottraendo 2 al valore corrente del link counter di DIR.