# Sistemi Operativi
# 18 settembre 2008
### Esame completo

Si risponda ai seguenti quesiti, giustificando le risposte.

1. 1. Si descrivano vantaggi e svantaggi dell'implementazione dei thread a livello di kernel e a livello utente.
   2. In quale caso e quali system call sono da evitare?

   **Risposta:**

   1. Nel caso dei thread a livello utente si ha un'efficienza maggiore in quanto tutte le operazioni sui thread vengono eseguite in user space, per mezzo di un'apposita libreria, mentre nel caso dei thread a livello kernel bisogna ricorrere ad opportune chiamate di sistema passando ogni volta da user mode a kernel mode (operazione dispendiosa). I thread a livello utente sono "invisibili" al sistema operativo che vede un unico processo. Quindi esiste una tabella dei thread separata per ogni processo e questo consente una maggiore flessibilità nel senso che, ad esempio, ogni processo può stabilire una politica di scheduling personalizzata per i propri thread. Lo svantaggio è che il blocco di un thread od il mancato rilascio della CPU da parte di quest'ultimo, porta al blocco dell'intero processo. Quando si parla di thread a livello kernel invece esiste un'unica tabella a livello di sistema per tutti i thread di tutti i processi. Quindi la politica di scheduling viene decisa a livello di sistema ed il blocco di un thread non è fatale per gli altri thread che vengono visti come entità autonome dal sistema operativo. In un sistema in cui i thread sono implementati a livello utente, un thread rilascia la CPU chiamando *thread_yield* quando, ad esempio, si mette in attesa di un risultato che deve essere prodotto da un altro thread, in quanto i thread stessi non sono visibili al sistema operativo. Quindi, senza richiamare *thread_yield* non vi potrebbe essere il rilascio della CPU a favore degli altri thread del processo.
   2. Per quanto detto al punto precedente, nel caso dei thread a livello utente bisogna evitare le system call bloccanti, dato che potrebbero portare al blocco dell'intero processo.

2. Si assuma una politica di scheduling della CPU basata su priorità dinamiche e con prelazione. La priorità di un processo cambia sommando un intero (positivo, negativo o nullo) che dipende dallo stato, come segue:

   $\alpha$ : cambiamento di priorità quando un processo diventa *running*
   $\beta$ : cambiamento di priorità quando un processo diventa *ready*
   $\gamma$ : cambiamento di priorità quando un processo sta eseguendo *operazioni di I/O*

   Un processo ha priorità 0 quando è creato. Un processo con valore di priorità maggiore ha una maggiore priorità di scheduling.

   Si discutano le politiche di scheduling nei seguenti casi:

   1. $\alpha > 0$, $\beta = 0$, $\gamma = 0$
   2. $\alpha = 0$, $\beta > 0$, $\gamma = 0$
   3. $\alpha = \beta = 0$, $\gamma > 0$
   4. $\alpha < 0$, $\beta = 0$, $\gamma = 0$

   Il comportamento dello scheduling cambia se la priorità di un processo viene riportata a 0 ogni volta che un processo, dopo essere stato schedulato, rilascia la CPU?

   **Risposta:** Per quanto riguarda le politiche di scheduling, abbiamo i casi seguenti:

   1. $\alpha > 0$, $\beta = 0$, $\gamma = 0$: in questo caso vengono favoriti i processi CPU-bound (più un processo viene eseguito, maggiore diventa la sua priorità) e si tende ad avere meno prelazione.
   2. $\alpha = 0$, $\beta > 0$, $\gamma = 0$: i processi in coda ready tendono ad aumentare la propria priorità: in questo modo diminuisce la probabilità di starvation.
   3. $\alpha = \beta = 0$, $\gamma > 0$: in questo caso vengono favoriti i processi I/O-bound.
   4. $\alpha < 0$, $\beta = 0$, $\gamma = 0$: è un meccanismo da evitare, perché abbassando la priorità di un processo appena schedulato si rischia che questo venga immediatamente prelazionato.

   Nel caso in cui il processo si veda riportare a zero la priorità ogni volta che viene eseguito, in pratica significa che la storia pregressa non conta e quindi anche processi nuovi (i.e., appena creati) non risultano sfavoriti rispetto a quelli che sono in esecuzione da più tempo.

# Sistemi Operativi
# 18 settembre 2008
**Esame completo**

3. Si descriva il funzionamento dell'algoritmo di rimpiazzamento delle pagine LRU. Quanti page fault si verificano sulla seguente stringa di riferimenti con l'algoritmo LRU, avendo a disposizione 4 frame fisici di memoria e 9 pagine?

   6 0 5 7 8 7 0 5 1 6 4 2 1 3

   **Risposta:** L'algoritmo LRU (Least Recently Used) "utilizza" il passato per "prevedere" il futuro, rimpiazzando la pagina che da più tempo non viene usata. Nel caso in esame si verificano 10 page fault come si vede dalla seguente simulazione:

| 6 | 0 | 5 | 7 | 8 | 7 | 0 | 5 | 1 | 6 | 4 | 2 | 1 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6 | 0 | 5 | 7 | 8 | 7 | 0 | 5 | 1 | 6 | 4 | 2 | 1 |
| | | 6 | 0 | 5 | 5 | 8 | 7 | 0 | 5 | 1 | 6 | 4 | 2 |
| | | | 6 | 0 | 0 | 5 | 8 | 7 | 0 | 5 | 1 | 6 | 4 |
| | | | | 6 | 6 | 6 | 6 | 8 | 7 | 0 | 5 | 5 | 6 |
| | | | | | | | | 6 | 8 | 7 | 0 | 0 | 5 |
| | | | | | | | | | | 8 | 7 | 7 | 0 |
| | | | | | | | | | | | 8 | 8 | 7 |
| | | | | | | | | | | | | | 8 |
| P | P | P | P | P | | | | P | P | P | P | | P |

4. Si illustrino le differenze fra le principali modalità di input-output:

   1. Programmed I/O (ovvero, I/O a interrogazione ciclica),
   2. Interrupt driven I/O (ovvero, I/O guidato dagli interrupt),
   3. DMA (ovvero, Direct Memory Access: trasferimento diretto in memoria).

   Si facciano delle considerazioni sull'efficienza delle varie modalità.

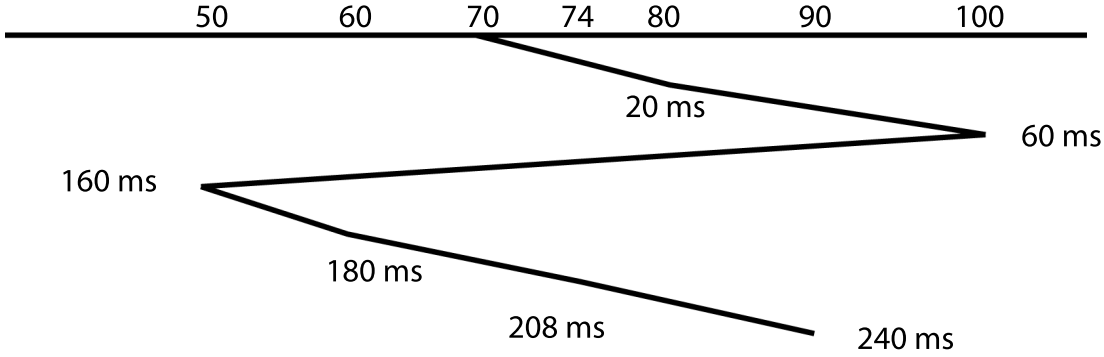
   **Risposta:** Mentre con la modalità Programmed I/O (I/O a interrogazione ciclica) il processore manda un comando di I/O e poi attende che l'operazione sia terminata, testando lo stato del dispositivo con un loop busy-wait (polling), con la modalità Interrupt-driven I/O, una volta inviato il comando di I/O, il processo viene sospeso fintanto che non arriva un interrupt a segnalare il completamento dell'operazione. Durante la sospensione del processo, la CPU può mandare in esecuzione altri processi o thread. Di fondamentale importanza è il vettore di interrupt che consente di selezionare la routine di gestione opportuna per ogni tipo di interrupt. Ovviamente la prima modalità è efficiente soltando nel caso in cui la velocità del dispositivo di I/O sia paragonabile a quella della CPU. La modalità DMA richiede un controller DMA e funziona in questo modo: la CPU imposta i registri del controller DMA specificando il tipo di azione di I/O, l'indirizzo di memoria ed il conteggio di byte da trasferire. Poi i dati vengono trasferiti senza più richiedere l'intervento della CPU; infatti il controller del dispositivo di I/O riceve le richieste di lettura o scrittura da parte del controller DMA a cui notifica il completamento dell'operazione una volta che ha trasferito il byte da/verso l'indirizzo di memoria corretto (specificato dal controller DMA). A questo punto il controller DMA incrementa l'indirizzo di memoria comunicandolo sul bus e decrementa il conteggio dei byte da traferire, ripetendo la richiesta di lettura o scrittura al controller del dispositivo fintanto che il conteggio dei byte non raggiungerà lo zero. Soltanto a questo punto verrà inviato un interrupt alla CPU che potrà far ripartire il processo sospeso. Siccome il controller DMA deve bloccare il bus per consentire i trasferimenti dal controller del dispositivo alla memoria, se anche la CPU ha bisogno di accedere al bus dovrà aspettare, venendo così rallentata.

5. Si consideri un disco gestito con politica C-LOOK. Inizialmente la testina è posizionata sul cilindro 70, direzione ascendente; lo spostamento ad una traccia adiacente richiede 2 ms. Al driver di tale disco arrivano richieste per i cilindri 80, 74, 100, 90, 50, 60 rispettivamente agli istanti 0 ms, 14 ms, 16 ms, 42 ms, 50 ms, 70 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le richieste in oggetto?

**Risposta:**

1. Le richieste vengono servite nell'ordine 80, 100, 50, 60, 74, 90 come illustrato dal seguente diagramma:




2. Il tempo di attesa medio è dato da $\frac{(20-0)+(60-16)+(160-50)+(180-70)+(208-14)+(240-42)}{6} = \frac{20+44+110+110+194+198}{6} = \frac{676}{6} = 112,67\ ms$.

6. Si descrivano brevemente i sette livelli del modello ISO/OSI.

**Risposta:**

**Fisico:** gestisce i dettagli fisici della trasmissione dello stream di bit.

**Strato data-link (collegamento dati):** gestisce i *frame*, parti di messaggi di dimensione fissa, nonché gli errori e recuperi dello strato fisico.

**Strato di rete:** fornisce le connessioni e instrada i pacchetti nella rete. Include la gestione degli indirizzi degli host e dei percorsi di instradamento.

**Trasporto:** è responsabile dell'accesso di basso livello alla rete e del trasferimento dei messaggi tra gli host. Include il partizionamento di messaggi in pacchetti, il riordino di pacchetti, e la generazione di indirizzi fisici.

**Sessione:** implementa le sessioni, ossia delle comunicazioni tra processi.

**Presentazione:** risolve le differenze tra i varî formati dei diversi siti (p.e., conversione di caratteri).

**Applicazione:** interagisce con l'utente: trasferimento di file, protocolli di login remoto, trasferimento di pagine web, e-mail, database distribuiti, ...

Il punteggio attribuito ai quesiti è il seguente: 6, 9, 4, 4, 5, 3. Totale 31 punti.