- 1. Che cosa si intende con il termine *thrashing*? Che conseguenze può avere?
  2. Descrivere il meccanismo del copy-on-write. Quando si applica e che vantaggio comporta?
  3. (*) Supponendo di avere un sistema con tre frame e sette pagine, addottando una politica di rimpiazzamento LRU, quanti page fault si verificheranno con la reference string seguente?

     3 4 0 1 0 2 4 5 1 2 6

     (Si assuma che i tre frame siano inizialmente vuoti.)

Risposta:

1. Quando la memoria fisica libera (e quindi il numero di frame liberi) è insufficiente a contenere il working set corrente di un processo, quest'ultimo comincerà presto a generare parecchi page fault, rallentando considerevolmente la propria velocità d'esecuzione. Quando parecchi processi cominciano ad andare in thrashing, ovvero a spendere più tempo per la paginazione che per l'esecuzione, il sistema operativo potrebbe erroneamente essere indotto a dedurre che sia necessario aumentare il grado di multiprogrammazione (dato che la CPU rimane per la maggior parte del tempo inattiva a causa dell'intensa attività di I/O). In questo modo vengono avviati nuovi processi che però, a causa della mancanza di frame liberi, cominceranno a loro volta ad andare in thrashing: in breve le prestazioni del sistema collassano fino ad indurre l'operatore a dover terminare forzatamente alcuni processi.
2. Il meccanismo del copy-on-write consente di condividere sia il codice che i dati a due processi. Ciò risulta utile in particolare in ambiente Unix in seguito ad una `fork` dato che sia il processo padre che il figlio inizialmente condividono sia il codice che i dati. Ciò avviene facendo in modo che le page table dei due processi puntino alle stesse pagine in memoria (in questo modo non è necessario duplicare le pagine dei dati al momento della `fork`). Tuttavia le pagine contenenti i dati vengono marcate come pagine di sola lettura. Quando uno dei due processi tenta di modificare una pagina dati, avviene una violazione della protezione di sola lettura e si solleva una trap al sistema operativo. La routine di gestione della trap a questo punto copia effettivamente la pagina ed aggiorna le page table in modo che ogni processo punti correttamente alla propria copia (viene anche tolto il flag di sola lettura ovviamente). In questo modo la dispendiosa attività di duplicare le pagine dei dati viene ritardata fino al primo tentativo di modifica da parte di uno dei processi coinvolti, facendo in modo che la `fork` venga eseguita molto velocemente.
3. Simuliamo il funzionamento di LRU:

	3	4	0	1	0	2	4	5	1	2	6
		3	4	0	1	0	2	4	5	1	2
			3	4	4	1	0	2	4	5	1
				3	3	4	1	0	2	4	5
						3	3	1	0	0	4
								3	3	3	0
											3
	P	P	P	P		P	P	P	P	P	P

Si verificano quindi dieci page fault.

- 1. Elencare i passi eseguiti dal *driver delle interruzioni*.
  2. Uno dei problemi con una struttura di directory a grafo è la presenza di cicli da evitare durante la visita del filesystem e la creazione di *garbage* in seguito ad operazioni di cancellazione. Si illustrino almeno due possibili soluzioni.
  3. Si definisca il concetto di *file* e si illustrino le operazioni di base su di esso.

  Risposta:

  1. I passi eseguiti dal driver delle interruzioni sono i seguenti:
     - salvare i registri della CPU,
     - impostare un contesto per la procedura di servizio (inizializzare TLB, MMU, stack ecc.),
     - inviare un segnale di *acknowledge* al controllore degli interrupt (per avere interrupt annidati),
     - copiare la copia dei registri nel PCB,
     - eseguire la procedura di servizio che accede al dispositivo,
     - eventualmente, cambiare lo stato a un processo in attesa (e chiamare lo scheduler di breve termine),
     - organizzare un contesto (TLB, MMU ecc.) per il processo successivo,
     - caricare i registri del nuovo processo dal suo PCB,
     - continuare l'esecuzione del processo selezionato.
  2. Alcune possibili soluzioni sono le seguenti:
     - permettere la creazione di link ai soli file (soluzione adottata in Unix per i link hard),
     - limitare il numero di link attraversabili (soluzione adottata in Unix per i link simbolici),
     - implementare un meccanismo di garbage collection che marchi durante la visita al filesystem le parti raggiungibili ed elimini tutto ciò che sfugge alla marcatura (bisogna fare attenzione a non visitare più volte gli stessi elementi durante la marcatura per non incappare negli eventuali cicli),
     - verificare la presenza di cicli al momento della creazione di nuovi link (soluzione costosa).

3. Un file è una collezione di informazioni correlate a cui viene dato un nome a che viene memorizzata nella memoria secondaria. Formalmente un file è un tipo di dato astratto su cui devono essere definite almeno le seguenti operazioni di base:
   - creazione,
   - cancellazione,
   - lettura,
   - scrittura,
   - riposizionamento,
   - troncamento.

- 1. Spiegare la differenza fra servizi *orientati alla connessione* e servizi *non orientati alla connessione*.
  2. Descrivere il funzionamento di una chiamata RPC.

  Risposta:

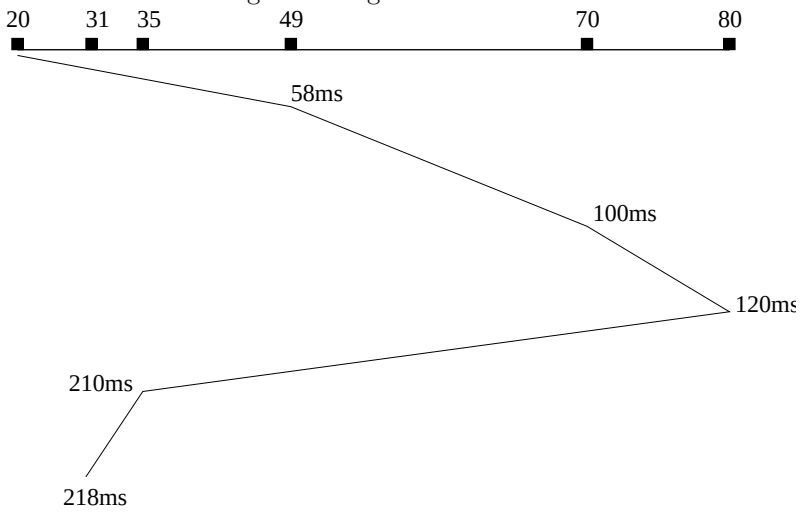
  1. I servizi orientati alla connessione stabiliscono una connessione, ovvero, un canale virtuale per tutta la durata della comunicazione in modo da assicurare il trasferimento dei dati in sequenza. Invece i servizi non orientati alla connessione trasferiscono singoli messaggi senza stabilire e mantenere una vera connessione. In particolare quindi il ricevente può ricevere i messaggi in ordine diverso da quello in cui sono stati spediti.
  2. Una Remote Procedure Call (RPC) consente ad un processo in esecuzione su un host di effettuare una chiamata ad una procedura che viene eseguita su un host remoto come se questa fosse una normale chiamata ad una funzione locale. Lo scambio di messaggi necessario per il funzionamento di una RPC è completamente invisibile al programmatore. Sostanzialmente quando un processo su un host A chiama una procedura su un host B, il processo chiamante su A viene sospeso, l'informazione necessaria per la computazione (i parametri della procedura) vengono comunicati via rete e l'esecuzione prosegue su B. Per effettuare una chiamata RPC il processo chiamante (client) utilizza una piccola procedura (client stub) che rappresenta la procedura remota nello spazio di indirizzamento del client. Il client stub organizza i parametri in un messaggio (marshaling) che viene spedito all'host remoto. Giunto a destinazione il messaggio, esso viene elaborato dal server stub che chiama la procedura responsabile della computazione. Il risultato viene poi rispedito via rete dal server stub al client stub che lo restituisce al processo chiamante. Quest'ultimo può quindi riprendere la sua esecuzione come in seguito ad una normale chiamata di procedura locale.

- (*) Si consideri un disco gestito con politica LOOK. Inizialmente la testina è posizionata sul cilindro 20, ascendente; lo spostamento ad una traccia adiacente richiede 2 ms. Al driver di tale disco arrivano richieste per i cilindri 70, 31, 49, 35, 80, rispettivamente agli istanti 0 ms, 30ms, 40 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 quattro richieste in oggetto?

Risposta:

1. L'ordine in cui vengono servite le richieste è 49, 70, 80, 35, 31 come illustrato dal seguente diagramma:



2. Il tempo d'attesa medio è $\frac{(58-40)+(100-0)+(120-70)+(210-50)+(218-30)}{5} = \frac{18+100+50+160+188}{5} = 516/5 = 103,2\ ms.$

Il punteggio attribuito ai quesiti è il seguente: 3, 3, 5, 3, 3, 3, 3, 3, 6 (totale: 32).