- Spiegare a cosa servono le chiamate di sistema (system call) in un sistema operativo.

  Risposta: Le chiamate di sistema (system call) fungono da interfaccia tra i programmi in esecuzione ed il sistema operativo: sono solitamente disponibili come speciali istruzioni assembler o come delle funzioni nei linguaggi che supportano direttamente la programmazione di sistema (ad esempio, il C). Esistono vari tipi di chiamate di sistema relative al controllo di processi, alla gestione dei file e dei dispositivi, all'ottenimento di informazioni di sistema, alle comunicazioni. L'invocazione di una chiamata di sistema serve ad ottenere un servizio dal sistema operativo; ciò viene fatto passando dal cosiddetto *user mode* al *kernel mode* per mezzo dell'istruzione speciale `TRAP`. Infatti, il codice relativo ai servizi del sistema operativo è eseguibile soltanto in kernel mode per ragioni di sicurezza. Una volta terminato il compito relativo alla particolare chiamata di sistema invocata, il controllo ritorna al processo chiamante passando dal kernel mode allo user mode.

- Spiegare la differenza fra processi e thread.

  Risposta: Un processo è un programma in esecuzione; quindi si tratta di un'entità attiva con un program counter ed il contenuto dei registri che rappresentano l'attività corrente, il proprio stack che contiene i dati temporanei (parametri di una procedura, indirizzi di ritorno, variabili locali), una sezione di codice, una sezione di dati, delle risorse allocate (file aperti ecc.) ed eventualmente uno heap per allocare della memoria dinamicamente. Nel caso più semplice ogni processo esegue un unico *flusso* di operazioni. In generale però un processo può contenere diversi flussi di esecuzione concorrenti detti thread. Quindi i thread rappresentano delle unità di esecuzione, all'interno dei processi, caratterizzate da un proprio valore del program counter e dei registri, da un proprio stack e stato di esecuzione. Tutte le altre risorse (codice, dati, risorse del sistema operativo) sono condivise dai thread appartenenti ad uno stesso processo.

- In coda ready arrivano i processi P_1, P_2, P_3, P_4, con CPU burst e istanti di arrivo specificati in tabella:

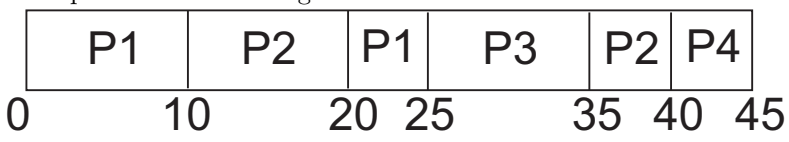
	arrivo	burst
P_1	0	15ms
P_2	5	15ms
P_3	15	10ms
P_4	30	5ms

  1. Se i processi terminano nell'ordine P_1, P_3, P_2, P_4, quale può essere l'algoritmo di scheduling? (Si trascuri il tempo di latenza del kernel.)
     (a) RR q=10ms
     (b) Scheduling non-preemptive
     (c) SJF
     (d) RR q=15ms
     (e) Nessuno dei precedenti

2. (*) Si calcoli il tempo di reazione medio per i processi P_1, P_2, P_3, P_4 della tabella sopra nel caso di algoritmo RR con q=10ms (si trascuri il tempo di latenza del kernel).

Risposta:

1. a), b), c)
2. Il diagramma di Gantt relativo all'algoritmo di scheduling RR con quanto=10ms è il seguente:



Pertanto il tempo di reazione medio è pari a $\frac{0+(10-5)+(25-15)+(40-30)}{4} = \frac{25}{4} = 6,25$

- (*) Si consideri un sistema con memoria paginata a due livelli, in cui sia la page table esterna che quella interna necessarie alla risoluzione dell'indirizzo siano mantenute in memoria principale. Il tempo di accesso alla memoria principale sia $t = 60ns$.

1. Qual è il tempo effettivo di accesso alla memoria (EAT: Effective Access Time)?
2. Aggiungendo un TLB (Translation Look-aside Buffer), con tempo di accesso $\epsilon = 1ns$, quale hit rate (α) dobbiamo avere per un degrado delle prestazioni del 5% rispetto a t?

Risposta:

1. Il tempo effettivo di accesso alla memoria è $3t$, ovvero, 180 ns; infatti sono necessari 60 ns per accedere alla page table esterna, 60 ns per accedere alla page table interna ed infine 60 ns per accedere alla locazione nel frame fisico in memoria.
2. Un degrado del 5% rispetto a t significa un EAT pari a $1,05 \cdot t$, ovvero, 63 ns. Quindi si ha quanto segue (α rappresenta l'hit rate):

$$\begin{array}{rcc} EAT & = & \epsilon + \alpha t + (1-\alpha)3t \\ 63 & = & 1 + 60\alpha + (1-\alpha) \cdot 180 \\ 63 & = & 181 - 120\alpha \end{array}$$

da cui si ricava $\alpha = \frac{118}{120} = 0,98$ (98%).

- Spiegare come funziona una tabella delle pagine invertita. Qual è lo svantaggio principale di questa tecnica nella traduzione da indirizzo virtuale a indirizzo fisico?

Risposta: L'idea alla base di questa tecnica è di mantenere una tabella con una entry per ogni frame fisico della memoria, non per ogni pagina virtuale. Le informazioni contenute in ogni entry sono il numero della pagina (virtuale) memorizzata in quel frame ed informazioni riguardanti il processo che possiede la pagina. In questo modo diminuisce la memoria

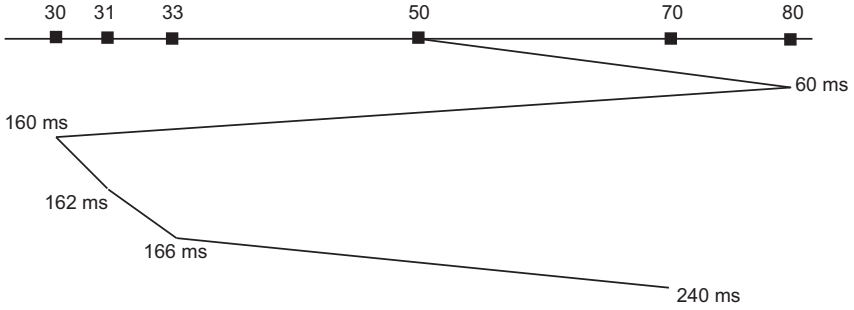
necessaria per memorizzare le page table, a discapito del tempo di accesso alla tabella stessa. Quest'ultimo ovviamente aumenta dato che, per tradurre un indirizzo virtuale nel corrispondente indirizzo fisico, è necessario scandire tutta la tabella alla ricerca di un'entry che contenga il numero di pagina virtuale che stiamo cercando di mappare. Questo è lo svantaggio principale di questa tecnica al momento di tradurre l'indirizzo virtuale nel corrispondente indirizzo fisico.

- Descrivere i passi eseguiti durante un trasferimento di dati da un dipositivo di I/O ad un buffer in memoria in un sistema *interrupt-driven* senza controller DMA.

  Risposta: Il device driver inizia l'operazione impostando il controller della periferica con i dati corretti. Il controller inizia a recuperare i dati dal dispositivo e, non appena questi sono pronti o si verifica un errore oppure si verifica il completamento dell'operazione di lettura, viene generato un interrupt. La CPU, quando riceve l'interrupt, trasferisce il controllo all'interrupt handler che processa i dati ricevuti dal controller, memorizzandoli nella posizione opportuna nel buffer in memoria. A questo punto la CPU restituisce il controllo al processo interrotto (eventualmente il processo che esegue il codice del device driver) e si prosegue fino al completamento dell'operazione di I/O. Siccome non c'è controller DMA possono intervenire parecchi interrupt (uno per ogni blocco di dati letti dal controller del dispositivo) prima che si giunga alla fine dell'operazione di I/O.

- (*) Si consideri un disco gestito con politica C-LOOK. Inizialmente la testina è posizionata sul cilindro 50 con moto in direzione ascendente; lo spostamento ad una traccia adiacente richiede 2 ms. Al driver di tale disco arrivano richieste per i cilindri 80, 30, 31, 70, 33, rispettivamente agli istanti 0 ms, 30ms, 40 ms, 50 ms, 70 ms. Si trascuri il tempo di latenza. In quale ordine vengono servite le richieste?

  Risposta: Le richieste vengono soddisfatte nell'ordine 80, 30, 31, 33, 70, come evidenziato nel diagramma seguente:



Il punteggio attribuito ai quesiti è il seguente: 3, 3, 3, 3, 3, 3, 4, 4, 6 (totale: 32).