- 1. In un sistema operativo multithread, quali risorse sono condivise tra thread dello stesso processo e quali sono private dei singoli thread?
  2. In quali situazioni è conveniente usare thread di livello utente? Si faccia qualche esempio.

  Risposta:

  1. I thread di uno stesso processo condividono le seguenti risorse:
     - spazio di indirizzamento,
     - variabili globali,
     - file aperti,
     - processi figli,
     - timer in scadenza,
     - segnali e routine di gestione dei segnali,
     - informazioni di accounting.

     Le risorse private ad ogni thread invece sono le seguenti:
     - il program counter (PC),
     - i registri,
     - lo stack,
     - lo stato di esecuzione.
  2. I thread a livello utente sono convenienti per processi CPU-bound, ovvero, per processi che non necessitino di un'attività di I/O intensiva (che bloccherebbe il processo con tutti i suoi thread) in cui sia possibile scomporre e parallelizzare il lavoro. Infatti con i thread a livello utente tutte le operazioni di manipolazione dei thread (creazione, cancellazione, context switch ecc.) sono gestite tramite librerie in spazio utente che non necessitano quindi di trap al kernel e che risultano molto veloci ed efficienti. Ad esempio in un word processor è utile far eseguire il task di correzione ortografica in un thread a livello utente separato, rispetto al thread che si occupa di effettuare il rendering a video del testo.

- 1. Quali sono le caratteristiche di un algoritmo di scheduling della CPU adatto a sistemi time-sharing?
  2. Un algoritmo di scheduling basato su priorità statica è adatto a sistemi time-sharing?

  Risposta:

  1. La caratteristica fondamentale di un algoritmo di scheduling della CPU adatto a sistemi time-sharing è la presenza di un meccanismo di prelazione; infatti, questa risulta l'unica garanzia di evitare che un processo monopolizzi (intenzionalmente o a causa di un bug) il processore, impedendo di fatto l'interattività con il sistema (caratteristica primaria di un sistema time-sharing). Le altre caratteristiche peculiari sono per questa categoria di sistemi sono:
     - minimizzare il tempo di risposta, ovvero, il tempo intercorrente fra il momento in cui si impartisce un comando ed il momento in cui si ottiene il risultato;

– assicurare una buona proporzionalità, ovvero, schedulare i processi in modo che un compito semplice non richieda un tempo ritenuto eccessivo per il suo completamento (mentre un compito complesso può ragionevolmente richiedere un tempo superiore).

2. Un algoritmo di scheduling basato su priorità statica non è adatto a sistemi time-sharing in quanto può portare a situazioni in cui l'esecuzione di processi interattivi, ma a bassa priorità sia ritardata eccessivamente (facendo percepire all'utente un tempo di risposta troppo alto e quindi un cattivo servizio) o si arrivi addirittura alla starvation.

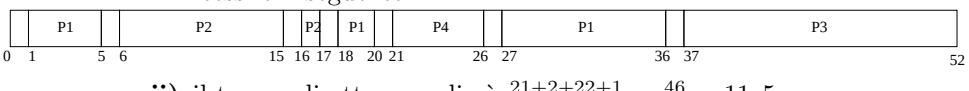
- In coda ready arrivano i processi P_1, P_2, P_3, P_4, con CPU burst e istanti di arrivo specificati in tabella:

	arrivo	burst
P_1	0	15ms
P_2	5	10ms
P_3	15	15ms
P_4	20	5ms

1. Se i processi terminano nell'ordine P_2, P_1, P_4, P_3, quale può essere l'algoritmo di scheduling? (Si trascuri il tempo di latenza del kernel e, nel caso di processi che arrivano in coda ready nello stesso istante, si dia priorità maggiore al processo con indice inferiore.)
   (a) RR q=10ms
   (b) RR q=5ms
   (c) Scheduling non-preemptive
   (d) SJF
   (e) Nessuno dei precedenti
2. (*) Si consideri ora tempo di latenza pari a 1ms e un algoritmo di scheduling SRTF. Si determini per i processi P_1, P_2, P_3, P_4 della tabella sopra:
   i) il diagramma di GANTT relativo all'esecuzione dei quattro processi;
   ii) il tempo di attesa medio;
   iii) il tempo di turnaround medio.

Risposta:

1. a), b).
2. Considerando un tempo di latenza pari a 1ms e un algoritmo di scheduling SRTF, abbiamo quanto segue:
   i) il diagramma di GANTT relativo all'esecuzione dei quattro processi è il seguente:



   ii) il tempo di attesa medio è $\frac{21+2+22+1}{4} = \frac{46}{4} = 11,5ms$,

iii) il tempo di turnaround medio è: $\frac{36+12+37+6}{4} = \frac{91}{4} = 22,75ms$.

- Dato un grafo di allocazione delle risorse contenente un ciclo, dire che cosa si può concludere in ognuno dei seguenti casi:

  1. c'è una sola istanza per ogni tipo di risorsa,
  2. ci sono più istanze per tipo di risorsa.

  Risposta:

  1. Nel caso in cui vi sia una sola istanza per tipo di risorsa, la presenza di un ciclo implica una situazione di deadlock.
  2. Nel caso invece in cui vi siano più istanze per tipo di risorsa, la presenza di un ciclo implica soltanto la *possibilità* che si verifichi una situazione di deadlock.

- Si illustri l'*algoritmo del Fornaio* per la risoluzione della sezione critica con n processi.

  Risposta: L'algoritmo del Fornaio permette di risolvere il problema della sezione critica con n processi nel modo seguente:

  – si utilizza uno schema di enumerazione che generi dei numeri in ordine crescente (eventualmente anche con ripetizioni);

  – ogni processo che desidera entrare nella sezione critica riceve un numero;

  – chi possiede il numero più basso entra nella sezione critica;

  – eventuali conflitti vengono risolti da un ordinamento statico dei processi: se P_i e P_j ricevono lo stesso numero e $i < j$, allora P_i ottiene l'accesso alla sezione critica prima di P_j (viceversa se $j < i$, allora P_j accede per primo alla sezione critica).

- Si consideri la seguente situazione, dove P_0, P_1, P_2 sono tre processi in esecuzione, C è la matrice delle risorse correntemente allocate, Max è la matrice del numero massimo di risorse assegnabili ad ogni processo e A è il vettore delle risorse disponibili:

	C			Max		
	A	B	C	A	B	C
P_0	0	2	0	0	5	1
P_1	0	0	0	2	4	3
P_2	2	3	0	2	4	2

Available (A)		
A	B	C
1	5	x

  1. Calcolare la matrice R delle richieste.
  2. Determinare il minimo valore di x tale che il sistema si trovi in uno stato sicuro.

  Risposta:

1. La matrice R delle richieste è data dalla differenza $Max - C$:

	R	
A	B	C
0	3	1
2	4	3
0	1	2

2. Se $x = 0$, allora non esiste nessuna riga R_i tale che $R_i \leq A$; quindi il sistema si trova in uno stato di deadlock. Se $x = 1$, allora l'unica riga di R minore o uguale a A è la prima. Quindi possiamo eseguire P_0 che, una volta terminato, restituisce le risorse ad esso allocate aggiornando A al valore $(1, 7, 1)$. A questo punto non esiste alcuna riga di R minore o uguale al vettore A e quindi il sistema è in stato di deadlock. Analogamente, se $x = 2$, dopo aver eseguito P_0, il valore di A è $(1, 7, 2)$. A questo punto si può eseguire P_2 aggiornando A al valore $(3, 10, 2)$: dato che l'unica riga di R rimasta da considerare non è minore o uguale al valore corrente di A, il sistema è in stato di deadlock.

   Il valore minimo di x per cui lo stato risulta sicuro è 3; infatti in questo caso esiste la sequenza sicura $\langle P_0, P_2, P_1 \rangle$. Dapprima si esegue P_0, generando il valore $(1, 7, 3)$ di A, poi si esegue P_2 portando A al valore $(3, 10, 3)$. A questo punto si conclude la sequenza eseguendo P_1 e generando il valore finale di A, ovvero, $(3, 10, 3)$.

Il punteggio attribuito ai quesiti è il seguente: 3, 3, 3, 3, 3, 6, 3, 3, 6 (totale: 33).