Quesito 1 (punti 6). Cinque processi *batch*, identificati dalle lettere $A - E$ rispettivamente, arrivano all'elaboratore agli istanti $0, 1, 3, 7, 8$ rispettivamente. Tali processi hanno un tempo di esecuzione stimato di $2, 7, 3, 4, 1$ unità di tempo rispettivamente. Per ognuna delle seguenti politiche di ordinamento:

1. Round Robin (divisione di tempo, senza priorità e con quanto di tempo di ampiezza 3)
2. Shortest Job First (nella variante "*Shortest Remaining Time Next*", ovvero con prerilascio)

determinare, trascurando i ritardi dovuti allo scambio di contesto: (i) il tempo medio di risposta; (ii) il tempo medio di attesa; (iii) il tempo medio di *turn-around*. I risultati finali possono essere espressi come frazioni. Nel caso di arrivi simultanei di processi allo stato di pronto, si dia la precedenza ai processi usciti dallo stato di esecuzione rispetto a quelli appena arrivati. Nel caso della politica Round Robin si assuma inoltre che i processi appena arrivati al sistema vengano inseriti in fondo alla coda dei pronti.

Quesito 2 (punti 5). Considerando i processi P1, P2, P3, P4, P5 e P6, in esecuzione su un elaboratore monoprocessore multiprogrammato, con l'ordine di arrivo e di esecuzione mostrato in figura 1, si determini quale/i tra le seguenti politiche di ordinamento senza uso di priorità esplicite possa(no) essere stata/e utilizzata/e:

1. First-In First-Out: indicare solo Sì o No
2. Round Robin: in caso di risposta affermativa, indicare un'ampiezza di quanto temporale concordante con l'ordinamento mostrato in figura 1
3. Shortest Job First (nella variante con prerilascio): indicare solo Sì o No.

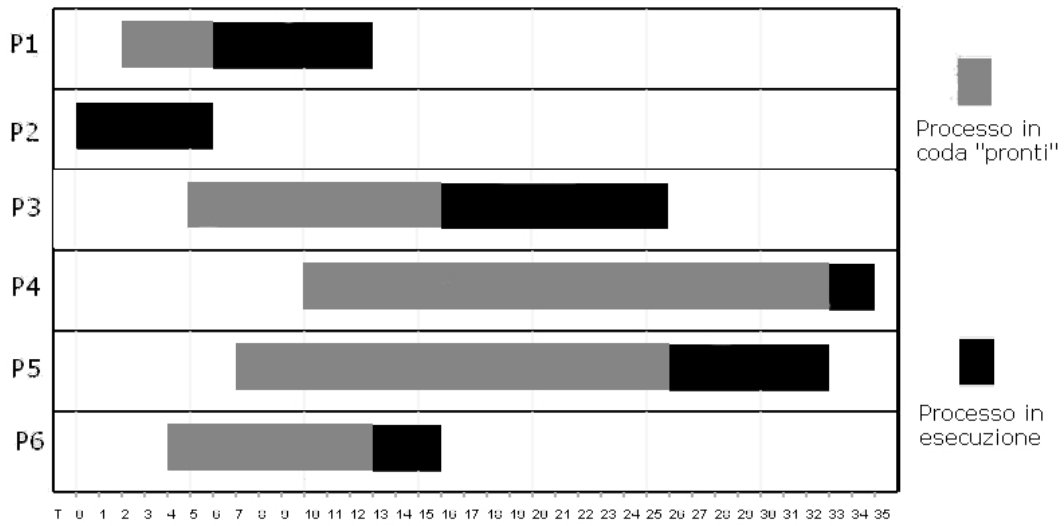

Figura 1: Ordine di arrivo e di esecuzione di sei processi su un elaboratore monoprocessore multiprogrammato.

Quesito 3 (punti 5). Discutere il posizionamento (quale prefisso), il significato e l'uso del campo "*Time to Live*" (TTL) utilizzato dalle strutture di messaggio impiegate in ambito TCP/IP.

Quesito 4 (punti 8). Illustrare il funzionamento della tecnica di assegnazione degli indirizzi IP denominata "*variable-length subnet mask*", evidenziandone i vantaggi rispetto alle tecniche alternative studiate a lezione, e fornire, discutendolo, un esempio d'uso.

Quesito 5 (punti 8). Un utente operante in ambiente GNU/Linux crea nel proprio spazio di lavoro due *file*, denominandoli rispettivamente "`./PARENT/pippo.txt`" e "`./PARENT/CHILD/pippo.txt`". Dopo aver scritto diverse quantità di testo nei due *file* usando il proprio editore preferito, su consiglio dell'amministratore del sistema, l'utente usa il comando `stat` per ottenere informazioni sui due *file* e sul *file system* che li ospita e le riporta diligentemente in tabella 1. Lo studente discuta il significato di tutte le voci ivi indicate e la correlazione tra loro e poi spieghi, con precisione e concisione, le ragioni per le quali l'editore di testo utilizzato non confonda i due *file* nonostante essi abbiano entrambi nome "`pippo.txt`".

Sul *file system*	
Namelength	255
Type	. . . ext3
Block size	4096
. . .	. . .
Blocks Total	13547642
Blocks Free	11640686
Blocks Available	10952502
Inodes Total	6884192
Inodes Free	6772305

Sui *file*		
	`./PARENT/pippo.txt`	`./PARENT/CHILD/pippo.txt`
Size	27	19
Blocks	16	8
Inode	2584677	2584676
Links	1	1
Access rights	. . .	. . .
Access time	2007-12-11 10:15:10. . . .	2007-12-11 10:15:25. . . .
Modify time	2007-12-11 10:15:07. . . .	2007-12-11 10:15:25. . . .
Change time	2007-12-11 10:15:10. . . .	2007-12-11 10:15:25. . . .

Tabella 1: Selezione di informazioni sui *file* e sul *file system* fornite dal comando `stat` nel caso in esame.

Soluzione 1 (punti 6).

- RR (con quanto di tempo di ampiezza 3)

```
processo A  AA                      LEGENDA DEI SIMBOLI
processo B  -bBBBbbbBBBbbbbB        - non ancora arrivato
processo C  ---ccCCC                x (minuscolo) attesa
processo D  -------ddddDDDddD       X (maiuscolo) esecuzione
processo E  --------eeeeeeE         . coda vuota

CPU         AABBBCCCBBBDDDEBD
coda        .b.ccbbbdddeeebd.
            .......deeebbbd..
```

processo	risposta	tempo di attesa	*turn-around*
A	0	0	0+2= 2
B	1	1+3+4= 8	8+7= 15
C	2	2	2+3= 5
D	4	4+2= 6	6+4= 10
E	6	6	6+1= 7
medie	2,60	4,40	7,80

- SJF_{SRTN} (ovvero con prerilascio)

```
processo A  AA                      LEGENDA DEI SIMBOLI
processo B  -bBbbbBbbbbbBBBBB       - non ancora arrivato
processo C  ---CCC                  x (minuscolo) attesa
processo D  -------DdDDD            X (maiuscolo) esecuzione
processo E  --------E               . coda vuota

CPU         AABCCCBDEDDDBBBBB
coda        .b.bbb.bdbbb.....
            ........b........
```

processo	risposta	tempo di attesa	*turn-around*
A	0	0	0+2= 2
B	1	1+3+5= 9	9+7= 16
C	0	0	0+3= 3
D	0	1	1+4= 5
E	0	0	0+1= 1
medie	0,20	2,00	5,40

Soluzione 2 (punti 5).

1. FIFO: Sì
2. RR: Sì, con qualsiasi quanto temporale di ampiezza maggiore o uguale al massimo tempo di esecuzione fra i processi considerati, ovvero 10 u.t. (per P3)
3. SRTN: No.

Soluzione 3 (punti 5). Il campo "*Time to Live*" si trova nell'intestazione del *datagram* e pertanto è in uso a livello IP (*network*). Il valore di questo campo (ampio 1 B), fissato alla creazione del *datagram* viene ispezionato e, se > 0, decrementato da ogni *router* cui il *datagram* pervenga. Quando il valore fosse sceso a 0 e il *datagram* non avesse ancora raggiunto la sottorete di destinazione, il *router* non lo propaga più in uscita, evitando in tal modo di congestionare la rete di interconnessione con messaggi che non abbiano ancora trovato "la via di casa".

Soluzione 4 (punti 8). Come sappiamo da lezione, la tecnica originaria per l'attribuzione degli indirizzzi IP (detta "*classful*" utilizza classi che ripartiscono i 32 *bit* dell'indirizzo in un campo rete e un campo nodo (*host*). Questa suddivisione causa un disastroso spreco di indirizzi perché la cardinalità effettiva dei nodi delle reti fisiche è in generale largamente inferiore a quella assegnata (certamente nel caso della Classe C e sovente in Classe B).
La prima tecnica correttiva storicamente proposta operava in modalità detta "*classless*" utilizza una maschera di interpretazione dell'indirizzo, detta *subnet mask* per "spostare" il confine tra parte di rete e parte di nodo, sottraendo *bit* a quest'ultima, in modo da ottenere cardinalità più rispondenti alle esigenze della rete fisica interessata.
La tecnica *classless* continuava però ad avere il difetto di ripartire la rete fisica data in sottoreti logiche di dimensioni uguali, per cui la suddivisione (e la maschera di sottorete) inevitabilmente dipendevano dalla sottorete logica più grande, continuando così a sprecare indirizzi.
La tecnica detta "*variable-length subnet mask*" (VLSM) risolve questo problema consentendo di utilizzare più maschere di sottorete entro lo spazio di indirizzamento di una singola rete fisica, in tal modo di fatto suddividendo le sottoreti eventualmente già esistenti nella rete fisica data ottenendo quindi un miglior utilizzo degli (scarsi) indirizzi disponibili.
Numerosi casi d'uso della tecnica VLSM sono discussi nei passati compiti d'esame. Si veda per esempio la soluzione del quesito 5 dell'appello del 18 settembre 2006.

Soluzione 5 (punti 8). Esaminiamo per prima la parte di tabella 1 che concerne il *file system*. Si tratta di un *file system* di tipo ext3, molto diffuso in ambiente GNU/Linux e, in quanto derivato del mondo UNIX, a struttura di allocazione a lista indicizzata basata su *i-node*. La dimensione del blocco è quella classica, ossia 4 KB. Conoscendo il numero totali di blocchi (per la cui indirizzazione servono 24 dei 32 *bit* della parola di architettura) possiamo facilmente derivare la dimensione della partizione, che è ampia 51,6 GB, occupata solo per il 14,1%. Risulta inoltre che 688.184 blocchi tra quelli liberi sono riservati dal Sistema Operativo per uso interno (per esempio per contenere puntatori a blocchi per gli *inode* con indici di prima, seconda e terza indirezione) e non sono dunque disponibili per i dati di utente. Il numero di *inode* rappresentabili nella partizione (6.884.192, esprimibile su 24 *bit*) deriviamo anche il massimo numero di *file* rappresentabili nella partizione (un *inode* per *file*). Dal numero di indici di *inode* liberi deriviamo anche che la partizione attualmente contiene 111.887 *file*.
Passiamo ora alle considerazioni concernenti i due *file* creati dall'utente evocato nel quesito. Si tratta di due *file* piuttosto piccoli quanto a contenuto: il primo consta di soli 27 B e il secondo 19 B. È importante però notare che, in ragione dell'algoritmo di assegnazione noto come *Buddy algorithm*, di blocchi contigui aggregati in gruppi chiamati regioni, al primo sono stati assegnati $16 = 2^4$ blocchi e al secondo $8 = 2^3$, in quantità notevolmente superiore al bisogno immediato, ma dimensionati sulle aspettivative presumibili di tipici *file* di utente. Ci viene fornito anche il numero d'ordine dell'*inode* corrispondente ai due *file*, che notiamo essere consecutivo. Questo può ragionevolmente suggerire che i due *file* siano stati creati l'uno appresso all'altro. Il campo informativo Links ci dice che entrambi i *file* hanno un solo collegamento diretto tra *directory* e *inode* corrispondente (ossia non sono attualmente riferiti da alcun *link* di tipo *hard*). Infine notiamo la differenza concettuale tra l'informazione relativa alla modifica, che concerne i dati utente del *file*, e al cambiamento, che invece concerne i metadati (che includono gli attributi e che ext3 salva prima nel *journal* e solo successivamente riflette nel *file* vero e proprio).
Per concludere, osserviamo che l'editore non può in alcun modo confondere i due *file* perché essi, essendo rappresentati da *inode* diversi, risultano al sistema come assolutamente diversi. Servirà infine osservare che la collocazione dei due *file* in *directory* diversi non ha alcuna influenza sulla garanzia di diversa identità tra essi, visto il *file system* in questione permette di creare *link* tra *file* ponendoli in posizioni qualunque nel *file system*.