Sistemi Operativi L-A

Compito di Martedì 18 Settembre 2007

CdS in Ingegneria Informatica - Prof. Paolo Bellavista

Compito B - Parte di Programmazione di Sistema (16 punti)

Si scriva un programma C che, utilizzando le System Call del sistema operativo UNIX, abbia un'interfaccia del tipo:

`terrorismo mailbox fileOut`

dove **mailbox** è un nome assoluto di file esistente nel file system mentre **`fileOut`** è il nome relativo del file che deve essere creato dal programma concorrente nel direttorio corrente. Dopo aver effettuato gli opportuni controlli sui parametri di invocazione, il processo iniziale **P0** deve generare altri 3 processi **P1**, **P2** e **P3**, con P1 figlio di **P0**, **P2** figlio di **P1** e **P3** figlio di **P2**. P3 svolge il ruolo di comandante di polizia e P0, P1 e P2 di agenti anti-terrorismo semplici.

Il file **mailbox** è stato ottenuto mettendo insieme le mailbox (ciascuna di 200 righe) di tre sospettati terroristi; ogni riga contiene "MittenteMsg : DestinatarioMsg : ContenutoMsg" e non può superare la lunghezza massima di 512 caratteri. L'agente di polizia **P0** deve cercare la parola "**terrorismo**" nelle prime 200 righe di **mailbox**. Per ogni riga in cui tale parola viene ritrovata, allora **P0** deve scrivere la riga incriminata su **fileOut** e comunicarla al comandante **P3**; inoltre, i mittente e destinatario di quella riga devono essere messi sotto osservazione. In particolare, P0 deve immediatamente comunicare all'agente P1 di cominciare a cercare quel mittente in tutte le righe della mailbox di pertinenza di P1 (da riga 200 a 399), in qualsiasi posizione della riga; analogamente P0 deve immediatamente comunicare all'agente P2 di cominciare a cercare la parola destinatario in tutte le righe della mailbox di pertinenza di P2 (da riga 400 a 599). Tutte le volte che P1 e P2 trovano righe contenenti, rispettivamente, mittente e destinatario sotto osservazione, i due processi devono anch'essi scrivere tali righe su **fileOut** e comunicarle al comandante **P3**.

In ogni istante, deve essere possibile per l'utente forzare la terminazione del programma concorrente premendo la combinazione di tasti <CTRL-C>. In tal caso, **P0** deve immediatamente terminare il suo lavoro, mentre **P1** e **P2** devono prima portare a termine la ricerca del mittente/destinatario in corso e poi terminare.

Si facciano le scelte di sincronizzazione dei processi ritenute più opportune, cercando di sequenzializzare il meno possibile le varie operazioni richieste.

Compito B - Parte di Programmazione Java Thread (8 punti)

Si scriva un programma che utilizzi i Java Thread per simulare una officina di revisione di autovetture. Nella officina può essere revisionata una sola autovettura per volta (thread Cliente.java). Ogni autovettura portata in officina deve sottostare ad un processo di revisione, costituito da quattro fasi, da

eseguirsi in ordine. Le fasi corrispondono, rispettivamente, alla accettazione, lavaggio, messa a punto e fatturazione al cliente.
Ogni fase del processo di revisione è condotta da un opportuno operatore (thread Operatore.java).
Per ogni autovettura entrata nell'autofficina, esiste una scheda di esecuzione lavori, che riporta il nome della vettura correntemente in revisione, e nella quale ogni operatore riporta lo stato di avanzamento dei lavori.
La prima fase di revisione consiste nella accettazione della vettura; sulla scheda della vettura viene appesa la stringa:

"*Accettazione vettura effettuata alle ore <timestamp>*".

La scheda così compilata viene lasciata a disposizione degli operatori seguenti per consentire i passi di revisione successivi.
Nella seconda fase, viene generato un timestamp ed appesa alla scheda la stringa:

"*Lavaggio vettura – durata: <durata> minuti*".

La scheda così compilata viene lasciata a disposizione degli operatori seguenti per consentire i passi di revisione successivi.
Nella terza fase, viene appesa alla scheda la stringa:

"*Messa a punto – durata: <durata> minuti*".

La scheda così compilata viene lasciata a disposizione dell'ultimo operatore. Nella quarta fase, viene appesa alla scheda la stringa:

"*Fatturazione – Totale: <spesa> euro*".

Al termine di questa fase, la scheda viene stampata a video e l'auto può lasciare l'officina.
I file allegati Launcher.txt, Operatore.txt e Cliente.txt contengono le implementazioni, rispettivamente, del programma principale, della classe Cliente e della classe Operatore. Inoltre, il file Officina.txt contiene uno scheletro della corrispondente classe. I candidati implementino i metodi mancanti *richiediRevisione()* ed *effettuaRevisione()* che servono a regolamentare l'accesso all'officina secondo le specifiche di sincronizzazione sopra illustrate.
N.B.: Per generare la stringa del timestamp si utilizzi il seguente codice:

```
Date now = new Date();
SimpleDateFormat sdf = new SimpleDateFormat("HH-mm-ss");
String timestamp = sdf.format(now);
```

Per generare le stringhe casuali *<durata>* e *<spesa>* si usi il seguente estratto di codice:

```
Random r = new Random();
int durata = r.nextInt(100);
...
```