Corso di Sistemi Operativi – Appello straordinario del 14 Novembre 2011

Si deve progettare un libreria per la stampa a schermo multithreaded e con priorità. La libreria/classe fornisce un unico metodo statico `stampa(String s, int prio)` dove `s` è una stringa da stampare a video e `prio` un valore di priorità che può valere 0 (alta priorità), 1 (media priorità), 2 (bassa priorità).

L'effetto dell'invocazione di `stampa()` per il thread chiamante deve essere quello di "prenotare" la visualizzazione a video della stringa `s`.

L'esecuzione della reale stampa a video deve essere asincrona: dunque, il codice di `stampa()` non deve essere bloccante per il thread chiamante, ma deve bufferizzare le richieste di stampa e uscire immediatamente, salvo il caso in cui le strutture dati che memorizzano tali richieste siano piene.

Un thread `Stampatore` si occupa di eseguire materialmente le stampe a video, prelevando continuamente le richieste di stampa dalle strutture dati che le memorizzano, e stampandole secondo le seguenti regole:

1. Una richiesta di stampa a bassa priorità può essere eseguita solo se non ci sono richieste ad alta e media priorità pendenti;
2. Una richiesta di stampa a media priorità può essere eseguita solo se non ci sono richieste ad alta priorità pendenti;
3. Ad eccezione della regola 1, le stampe a bassa priorità devono essere comunque eseguite al più presto se sono state eseguite più di 10 stampe consecutive a priorità 0 e 1 senza che avvenga alcuna stampa a priorità 2.
4. Ad eccezione della regola 2, le stampe a media priorità devono essere comunque eseguite al più presto se sono state eseguite più di 5 stampe consecutive a priorità 0 senza che avvenga alcuna stampa a priorità 1.
5. In assenza di richieste di stampa pendenti, il thread stampatore si pone in stato di attesa.

La libreria deve essere implementata garantendo **la mutua esclusione** nell'accesso ai dati condivisi **solo ove necessario**, al fine di garantire il massimo grado di parallelismo;

E' parte integrante della prova di esame completare le specifiche date nei punti non esplicitamente definiti, introducendo tutte le strutture dati che si ritengono necessarie, e risolvendo eventuali ambiguità. Non è consentito modificare il prototipo dei metodi se questo è stato fornito.

La prova può essere svolta in qualsiasi linguaggio di programmazione a scelta dotato di costrutti di supporto alla programmazione multi-threaded (esempio, C++ con libreria JTC, Java). Nel caso dovessero servire al candidato, si può assumere di avere a disposizione i costrutti di sincronizzazione `lock`, `read/write lock`, `barriera` e `buffer sincronizzato`. Non è pertanto necessario fornire l'implementazione di tali classi, ma solo eventuali loro specializzazioni ottenute per ereditarietà.

Non è esplicitamente richiesto di scrivere un `main()` o di implementare esplicitamente dei thread che invocano `stampa()`.

Alcuni suggerimenti per lo svolgimento (aggiunti dopo l'esame dal professore)

L'esercizio può essere risolto progettando una classe `Stampa` che prevede il metodo pubblico `'stampa'` (con prototipo definito in accordo alle specifiche) e il metodo privato (accessibile solo dalla classe `thread Stampatore`) `'prelevaStampa'`, il quale ritorna una stringa. La classe sarà dotata di tre buffer FIFO (non sincronizzati) che chiameremo `C0`, `C1` e `C2`, e un lock (o un monitor) che chiameremo `L`. La funzione `stampa(s,p)` è implementata:

1. Acquisendo il lock/monitor `L`;
2. Inserendo `s` ("put(s)") sul rispettivo buffer (priorità 0 = put sulla coda `C0`, priorità 1 = put sulla coda `C1`, priorità `C2` = put sulla coda `C2`, in generale priorità `i` → put sulla coda `C[i]`). Se questa operazione non si può concludere poiché il corrispondente buffer è pieno, ci si pone in `wait()` sul monitor `L` (o se `L` è un lock, su una condition opportuna legata a `L`);
3. Al termine della put è bene "notify"-care il thread stampatore della presenza di una nuova richiesta di stampa (nel caso in cui lo stampatore sia in attesa);
4. Rilasciando il lock/monitor `L`.

Il thread `Stampatore` si comporta nel seguente modo:

1. `S = prelevaStampa()`
2. Stampa `S`.
3. Ritorna al punto 1.

Il corpo della funzione `prelevaStampa()` conterrà le operazioni:

1. Acquisizione del lock `L`;
2. Estrazione di una stringa `S` da uno dei tre buffer. La scelta della stringa dipende dalle regole 1,2,3 e 4. In assenza di valori da stampare, lo Stampatore si pone in `wait()`;
3. Rilascio del lock `L`;