Relazione progetto di Sistemi Operativi:

Docente del corso: Patrizia Scandurra

LABORATORIO DI INFORMATICA

Sincronizzazione dei thread per l'accesso alla risorsa condivisa

Dandis Iana (1065350), 2019

1. Introduzione e breve descrizione del problema:

Il problema consiste nell'implementazione di un codice in linguaggio Java utilizzando il metodo di sincronizzazione (in)diretta in grado di gestire un laboratorio di informatica. In particolare, il laboratorio è aperto a un numero illimitato di studenti e dispone di un numero limitato di stazioni di lavoro. Ogni studente può usare in modo esclusivo e per un tempo finito da una a due stazioni di lavoro. Se le stazioni non sono disponibili lo studente deve attendere. In base alle necessità (una o due stazioni) gli studenti sono classificati in due code di attesa diverse:

1. Coda S: Per gli studenti che necessitano di utilizzare una sola stazione per volta
2. Coda T: Per gli studenti che necessitano di utilizzare due stazioni per volta

Le stazioni libere vengono assegnate con priorità agli studenti della coda T, in particolare i criteri di assegnazione sono:

- Se la coda T è vuota le stazioni disponibili vengono assegnate agli studenti presenti in coda S.
- Se la coda T non è vuota e ci sono almeno 2 stazioni disponibili, vengono assegnate le stazioni disponibili agli studenti presenti in coda T (due alla volta) finché possibile.
- Se non ci sono almeno 2 stazioni disponibili, allora non viene assegnata nessuna stazione di lavoro.

2. Gli aspetti più importanti della soluzione:

2a. Struttura del codice:

1. **Gli studenti** sono i thread dello stesso processo. Sono suddivisi in due tipologie per rispettare il vincolo della priorità in base alla richiesta del singolo studente:
   - **Tipologia 0**: corrisponde agli studenti che nell'assenza di postazione sono allocati nella coda S
   - **Tipologia 1**: corrisponde agli studenti che nell'assenza di postazione sono allocati nella coda T

2. **l'oggetto condiviso** è il laboratorio con le sue risorse limitate (stazioni di lavoro)
   quindi per l'accesso esclusivo da parte dei thread all'oggetto condiviso sono stati usati i metodi synchronized, mentre per la loro cooperazione i metodi **wait()** e **notifyAll()**

   - Synchronized **richiedePostazione** per la richiesta delle stazioni di lavoro da parte degli studenti
     La richiesta viene effettuata in base alla tipologia dello studente cosi da rispettare i vincoli imposti dal problema. Viene invocato il metodo **wait()** che mette i thread nelle rispettive code di attesa ogni qual volta non sono rispettate le condizioni di assegnazione:
     In particolare per agli studenti di **Tipologia 0** che richiedono una sola postazione: viene soddisfatta la sua richiesta solo nel caso in cui non c'è nessun studente di Tipologia 1 in attesa per occupare una stazione e le stazioni libere sono un numero maggiore di 2. Agli Studenti di **Tipologia 1** vengono assegnate le due postazioni ogni volta che sono disponibili le due risorse

   - Synchronized **rilasciaPostazione** per liberare il posto di lavoro e renderlo ancora disponibile per un altro studente invocando il metodo **notifyAll()** che risveglia tutti i thread nelle code di attesa mettendoli nel entry set garantendo in questo modo il progresso

2b. Suddivisione in file:

Il progetto è suddiviso in tre file:

1) Il primo file **"Laboratorio"** contiene: il costruttore dell'oggetto condiviso e i vari metodi di sincronizzazione descritti precedentemente
2) Secondo file **"Studenti"** il costruttore degli studenti e i metodi che invocano i thread sull'ogetto condiviso. In particolare un thread studente richiede una postazione di studio in base alla sua tipologia, usa e poi rilascia la risorsa. Per simulare l'utilizzo della stazione di lavoro viene invocato il metodo statico **sleep** con la funzione di mettere in attesa il thread corrente per un dato numero di millisecondi. Per stabilire il tempo di secondi si usa la funzione random in questo caso un tempo scelto da 0 a 5 secondi.
3) Terzo file **"Test"** contiene il metodo main per la creazione dei thread Studenti tramite un ciclo for e internamente al ciclo usa la funzione random per stabilire casualmente la tipologia dello studente 0 o 1.

3. Avvio ed esecuzione del programma

All'avvio del programma l'utente deve inserire da tastiera il numero delle stazioni che contiene il laboratorio ciò è possibile grazie all'importo della libreria **prog.io.ConsoleInputManager.**

Una volta inserito il numero delle stazioni il programma si avvia e procede all'infinito.

4. Conclusioni

In questo modo usando la sincronizzazione (in)diretta il progetto riporta la soluzione per la gestione del laboratorio di informatica implementata a livello di linguaggio (no package java.util.concurrent). Tuttavia per i vincoli imposti della condizione il programma non mitiga il fenomeno della starvation per i thread localizzati nella coda S. Questo problema si presenta in quanto il programma deve rispettare il vincolo di garantire un'assoluta priorità agli thread della coda T.