Relazione Progetto

Sistemi Operativi Laboratorio

Versione progetto: completa

Simulazione Multi-threaded, Multi-processo di un supermercato

Ludovico Venturi

Corso B
Matricola *578033*

Indice

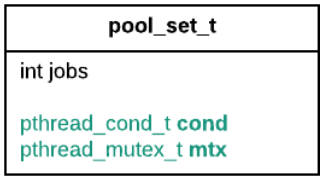
Docente di riferimento: **Massimo Torquati**

Università di Pisa
Dipartimeno di Informatica
Corso di Laurea in Informatica L-31

15 Luglio 2020

1 Pool

I clienti ed i cassieri vengono implementati come thread sempre 'vivi' all'interno del programma: vengono creati inizialmente K cassieri e C clienti e non ne verrano spawnati dinamicamente altri. Quando un cliente esce dal supermercato non viene terminato, esso va in attesa su una variabile di condizione. Per i cassieri il discorso è analogo, se la loro cassa è chiusa, attendono.
Per garantire il funzionamento di questo metodo ho strutturato i thread relativi ai clienti e ai cassieri utilizzando due *thread pool*.
Li gestisco attraverso una struttura dati da me definita di tipo *pool_ set_ t* che contiene oltre ad una lock ed una condition variable, il contatore di *jobs* disponibili.
Prima di eseguire il proprio lavoro un thread (cassiere o cliente) deve controllare che sia dispoinibile un lavoro: se *jobs* > *0* allora il thread prende il lavoro e decrementa *jobs* di 1.
Se invece *jobs* == *0* il thread si mette in attesa sulla condition variable del *pool_ set_ t*.



1.1 Concorrenza

Chiaramente la variabile *jobs* contenuta in *pool_ set_ t* genera una race condition:

- **jobs** letta e scritta da tutti i thread del pool di appartenenza; inizializzata e scritta dal *manager*

Per accedervi è necessario acquisire la *lock* sulla mutex relativa al pool_set. Le variabili di pool nel programma sono 2: una per i cassieri ed una per i clienti. Esse sono scollegate fra loro infatti i cassieri controlleranno solo la loro variabile di pool e viceversa.
Il *manager* può modificare *jobs* dei 2 pool nel seguente modo:

- **clienti** ogni E clienti che escono dal supermercato vengono resi disponibili E lavori (*jobs* += *E*)
- **casse** quando riceve la comunicazione dal direttore di aprire una cassa, incrementa di 1 *jobs*

2 Manager

2.1 Comunicazione con i Clienti

Il manager gestisce le comunicazioni dai clienti. Essi comunicano la loro entrata, la loro uscita e l'eventuale richiesta del permesso di uscita nel caso in cui non abbiano effettuato acquisti.
Tale comunicazione è gestita con una *unnamed pipe* aperta da entrambe le estremità in ogni thread.

2.1.1 Concorrenza

- **pipe, fd[0]** l'unico lettore è il manager, non vi è concorrenza su tale fd
- **pipe, fd[1]** possono scrivere sulla pipe tutti i C clienti ed anche il Thread Signal Handler

La comunicazione avviene seguendo un protocollo: viene inviato un messaggio di tipo *pipe_ msg_ code_ t* (mostrato in figura 1)

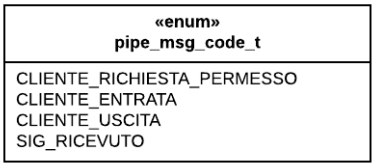


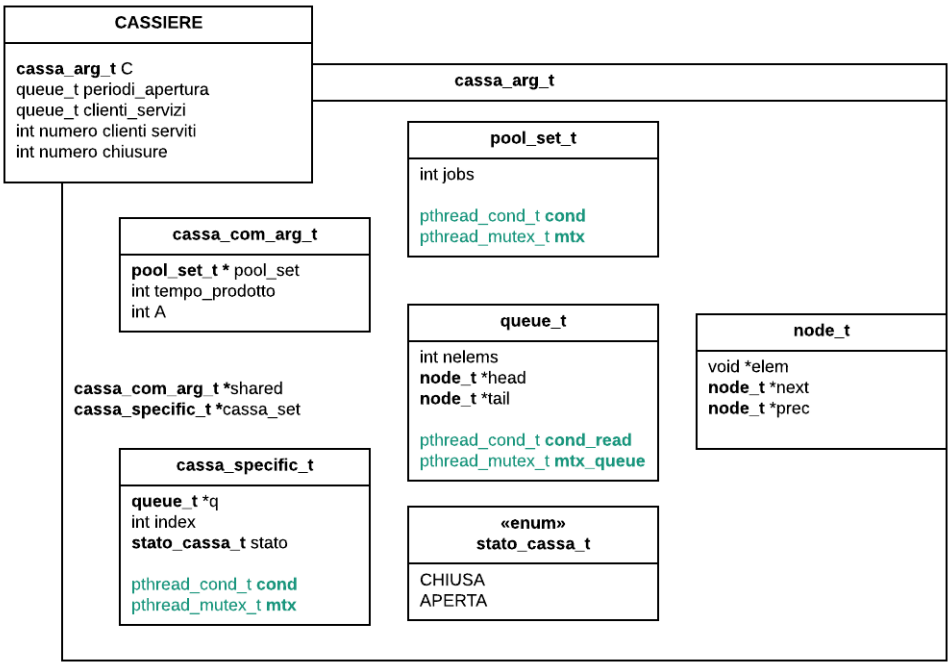
Figura 1: pipe_msg_code_t

3 Cliente

3.1 Concorrenza

- elem: **stato_attesa**
- **permesso_uscita** scritto solamente dal Manager e letto dal rispettivo cliente; reinizializzato dal cliente
  => va acquisita la risorsa attraverso *lock(cond_cliente)*

4 Cassiere



shared è condiviso fra tutte le casse mentre *cassa_set* è specifico per ogni cassa la scelta di usare un'ulteriore struttura dati è per incapsulamente e riservatezza dei dati delle che dovranno essere letti dai clienti

4.1 Concorrenza

Nel cassiere le variabili accedute in lettura e scrittura da più thread sono:

- q
- stato

5 Direttore

5.1 Comunicazione Direttore-Supermercato

6 Utilizzo