UNIVERSITÀ DI PISA

Sistemi Operativi e Laboratorio Corso A

Relazione progetto

Anno Accademico 2019/2020

Professore:

Giuseppe Prencipe

Presentato da:

Edoardo Ermini

Struttura del progetto

Il programma è diviso in 8 file principali *cashier.c, client.c, director.c, supermarket.c* contenuti nella directory *src* aventi i rispettivi header files all'interno della directory *include*. Oltre a questi files ce ne sono altri 4: *tsqueue.c* e *linkedlist.c* in *src* e i loro rispettivi header files all'interno di *include*.

cashier

cashier.h

Il file *include/cashier.h* è incluso in *include/client.h* che a sua volta è incluso in *src/director.c*, in quanto contiene le strutture dati tali da permettere al direttore di analizzare le informazioni dei clienti che i cassieri servono. Al suo interno troviamo:

- **struct analytics_data** utilizzata per mandare il numero dei clienti in coda ad una cassa al direttore
- **struct cashier_info** utilizzata per salvare le informazioni dei cassieri durante il servizio
- **client_info** utilizzata per salvare le informazioni di ogni cliente servito

cashier.c

Contiene 2 funzioni principali *cashier* e *send_analytics*, entrambe corrispondono a thread all'interno del processo, il primo avviato dal direttore (*src/director.c*) e il secondo avviato dal cassiere. Il thread *send_analytics* informa il direttore a intervalli regolari del numero dei clienti in attesa in coda, il thread *cashier* gestisce i clienti in coda, salva le informazioni dei clienti serviti e aggiorna le sue informazioni.

client

client.c

Contiene il thread *client* avviato dal direttore

director

director.c

Contiene 3 thread principali: *director, clients_handler, cashiers_handler*, il primo avviato dal processo *supermarket* gli altri 2 avviati dal primo. Il thread *clients_handler* gestisce l'entrata contingentata dei clienti, invece *cashiers_handler* gestisce l'apertura e la chiusura dei cassieri in base ai dati inviati da questi ultimi attraverso il thread *send_analytics*.

Scelte Implementative

fifo_tsqueue

È stata implementata una coda fifo thread safe con lock e condition variable utilizzata principalmente per simulare i clienti in coda alla cassa, ma data la sua versatilità in quanto permette l'inserimento di valori generici oltre che dai thread *client* e *cashier* è utilizzata anche dai thread *send_analytics*, *cashiers_handler* e *clients_handler*.
Nel file *src/tsqueue.c* c'è l'implementazione delle funzioni utilizzate per gestire la struttura dati **fifo_tsqueue_t** definita in *include/tsqueue.h*.

linkedlist

È stata implementata una lista collegata utilizzata principalmente per salvare le informazioni dei cassieri, in particolare i tempi di apertura per ogni apertura della cassa e i tempi di servizio per ogni cliente servito.
L'implementazione consiste in una struttura dati **int_list_node** definita in *include/linkdelist.h* e da varie procedure per operare con questa struttura dati, trattandola come se fosse un iteratore nella OOP.

client

Il meccanismo di attesa e avanzamento dei clienti ad una cassa è basato sull'utilizzo della coda fifo thread safe **cash_q**, definita in *include/cashier.h*, e di una pipe senza nome.
L'azione di mettersi in coda è stata implementata accodando alla coda **cash_q** il file descriptor di scrittura di una pipe creata precedentemente dal ciente.
Una volta fatto questo il thread *client* si mette in attesa chiamando la funzione bloccante read sul file descriptor di lettura della pipe, i messaggi che possono arrivare dal cassiere sono 2 definiti con 2 macro in *include/cashier.h*:

- **CLOSING (0)** viene inviato quando il cassiere sta chiudendo e costringe i clienti a cambiare cassa
- **NEXT (1)** viene inviato quando il cassiere comunica al cliente che deve uscire dalla coda per essere servito

Se il messaggio ricevuto è 1 i clienti inviano le proprie informazioni ai cassieri attraverso il buffer da una posizione **buff** definito in *include/cashier.h* e terminano.
La gestione dei clienti con 0 prodotti è basata, allo stesso modo, sull'utilizzo di un'altra coda fifo thread safe **zero_products_q** definita in *include/client.h* e di una pipe senza nome.
L'azione di notificare al direttore che si vuole uscire in quanto non si faranno acquisti è stata implementata inserendo in coda **zero_products_q** il file descriptor di scrittura di una pipe creata precedentemente dal cliente. Una volta fatto questo come descrito sopra il cliente si mettera in attesa sulla read dell'unico messaggio che può arrivare dal direttore: **D_EXIT_MESSAGE** definito in *include/director.h*. Non appena riceve il messagio invia le sue informazioni al direttore attraveso il buffer da una posizione **dir_buff** definito in *include/client.h* e termina.
Dato che ogni processo normalmente può avere aperti contemporaneamente al più 1024 file descriptor, si limita il numero di pipe aperte dai vari clienti nello stesso momento a 500.

cashier

È stata fatta la scelta di implementare l'azione da parte dei cassieri di informare il direttore del numero dei clienti in attesa in coda utilizzando il thread *send_analytics*.
send_analytics viene creato dal cassiere, utilizza una *nanosleep* per attendere il giusto lasso di tempo tra un invio e l'altro e per inviare le informazioni al direttore fa uso della struttura dati **analytics_data** definita in *include/cashier.h*, della coda thread safe **fifo_tsqueue** e dei metodi che la riguardano dichiarati e definiti in *include/tsqueue.h* e *src/tsqueue.c*.

director

Il compito del direttore di gestire i clienti e i cassieri è stato implementato utilizzando 2 thread figli del thread *director*:

- *cashier_handler*
- *clients_handler*

Prima di terminare il thread prende le informazioni dei cassieri e dei clienti salvate durante il periodo di apertura e le scrive all'interno del file di log indicato nel file di configurazione.

clients_handler

Se nessuna delle 2 variabili **quit** e **closing** è settata a 1 il thread crea e thread clienti ogni volta che terminano $c - e$ assegnando a ogni cliente creato un id che viene incrementato di 1 ad ogni thread cliente creato. Oltre a creare nuovi clienti il thread clients_handler gestisce i clienti con zero prodotti utilizzando la coda fifo thread safe **zero_products_q** facendoli terminare una volta ricevute le loro informazioni.
Quando si riceve un segnale di uscita (*quit = 1*) o di chiusura (*closing = 1*) Il thread non fa entrare piu clienti, attende che quelli attivi terminino facendo uscire anche i clienti con 0 prodotti e poi termina a sua volta.

cashiers_handler

Il thread apre e/o chiude le casse in base ai dati ricevuti dal thread *send_analytics*.
L'apertura e la chiusura corrispondono alla modifica della variabile condivisa **state** definita in *include/cashier.h*.
Quando il direttore chiude una cassa setta *state[i] = 0* con i l'id del cassiere e il cassiere i dato che controlla periodicamente se il direttore ha chiuso la cassa notifichera ai clienti in coda di cambiare cassa e terminerà.
Quando il direttore apre una cassa setta invece *state[i] = 1* e crea il thread cassiere con id i precedentemente terminato.
Quando si riceve un segnale di chiusura (*closing = 1*) il thread lavora normalmente fino a quando ci sono clienti attivi, una volta serviti tutti i clienti termina.
Quando si riceve un segnale di uscita (*quit = 1*) il thread chiude tutti i cassieri, attende che tutti i thread cashier terminino e poi termina a sua volta.

S1

Per capire quando la soglia s1 (il numero di casse con al più un cliente in coda oltre il quale si deve chiudere una cassa) viene superata si utilizza l'array *one_client*, aggiornato ad ogni nuovo dato ricevuto, e la procedura *cashiers_with_one_c* che prende l'array e restituisce il numero di clienti con al più un cliente. Quando la soglia viene superata si sceglie una delle casse con al piu un cliente in coda e viene chiusa cosi da evitare di chiudere una cassa che sta lavorando a regime e evitare lo spostamento di piu clienti.

S2

Quando la soglia s2 (il numero di clienti in coda ad una cassa oltre i quali se ne apre un'altra) viene superata si scorrono tutte le casse partendo dalla prima e non appena si trova una cassa chiusa viene aperta dal direttore.

supermarket

Esegue 3 operazioni principali:

1. Legge il file di configurazione
2. Fa partire il thread *director*
3. Gestisce i segnali in arrivo

La gestione dei segnali è custom solo per i segnali SIGHUP e SIGQUIT, quando riceve il primo setta la variabile *closing* definita in *include/cashier.h* a 1, quando riceve il secondo setta la variabile *quit* definita anch'essa in *include/cashier.h* a 1.

Compilazione ed esecuzione

All'interno della directory contenente il *Makefile* digitare:

```
$ make
$ make test
```

Si consiglia di eseguire make test con il terminale a schermo intero cosi da evitare che l'output non venga formattato correttamente.