Relazione progetto “Supermercato”
Sistemi Operativi e Laboratorio A.A. 2019/2020
Studente: Gaetano Barresi
Matricola: 579102

1. Progettazione: entità e loro comportamento

Il primo passo per la realizzazione del progetto è stato quello di creare un modello comportamentale per ogni entità presente nel supermercato.
Oltre al processo supermercato, in esecuzione saranno presenti:
- un thread “direttore” con tre thread di supporto;
- fino a K thread “cassiere” attivi contemporaneamente, con un thread di supporto ciascuno;
- fino a C thread “cliente” attivi contemporaneamente.

Di seguito sono illustrate le funzionalità del processo e dei thread.

1.1 Processo “supermercato”

Il processo supermercato, una volta avviato, controlla che il numero di argomenti sia corretto (1 argomento max) e che sia anche il file di configurazione corretto (conf.txt). Avvia successivamente il thread “direttore” e ne aspetta la sua conclusione prima di aggiornare il file resoconto.log con i dati acquisiti dalle casse, per poi terminare.

1.2 Thread “direttore”

Una volta avviato, il thread direttore avvia in ordine: un thread “cassiere”, (il supermercato apre all’inizio sempre con una sola cassa), il thread di supporto “controllo_casse”, C thread “cliente”, un thread di supporto “controllo_clienti_P0” e un thread di supporto “controllo_clienti_E”. Dopodiché si mette in attesa del segnale d’uscita, per poi attendere che tutte le casse vengano chiuse e infine termina.

1.2.1 Thread “controllo_casse”

Il thread controllo_casse consiste in una routine che permette la comunicazione ad intervalli regolari con i thread di supporto delle casse attive. Esso si mette in attesa di un segnale e una volta svegliato, controlla che le soglie S1 e S2 non siano state superate, altrimenti le gestisce rispettivamente chiudendo una cassa o aprendone una nuova (rispettando i limiti nella specifica del progetto). Termina con l’arrivo del segnale d’uscita

1.2.2 Thread “controllo_clienti_P0”

Questo thread si occupa di gestire i clienti che non acquistano e devono chiedere il permesso d’uscita al direttore prima di lasciare il supermercato. Funziona in modo analogo al thread “controllo_casse”: resta in attesa di un segnale da parte del cliente che intende uscire, gli dà il permesso e poi torna in attesa. Termina con l’arrivo del segnale di uscita.

1.2.3 Thread “controllo_clienti_E”

Costantemente controlla il numero dei clienti effettivamente presenti nel supermercato. Non appena questo numero arriva ad essere <= a C-E, questo processo fa entrare altri E clienti all’interno del supermercato. Termina all’arrivo del segnale d’uscita.

1.3 Thread “cassiere”

Il thread “cassiere” all’avvio ottiene la sua coda di clienti da servire e avvia il thread di supporto “aggiornamenti”. Successivamente entra in un ciclo while(true) interrotto soltanto dai segnali di uscita o dal segnale di chiusura proveniente dal direttore. Ad ogni ciclo serve un cliente appartenente alla sua coda, rispettando la politica FIFO. In caso di chiusura da parte del direttore, notifica tutti i clienti della propria coda la chiusura prossima e li invita a scegliere una nuova cassa. Interrotto il ciclo while, deposita i dati collezionati relativi alla propria cassa e se è l’ultimo thread “cassiere” a chiudere, notifica il thread “direttore”.

1.3.1 Thread “aggiornamenti”

Questo è il thread che comunica con “controllo_casse”. Periodicamente (“A” millisecondi, valore preso da conf.txt) invia un segnale al thread di controllo per svegliarlo. Termina con i segnali di chiusura o quando il direttore chiude la cassa a cui appartiene.

1.4 Thread “cliente”

Questo thread acquisisce i dati relativi al cliente, sceglie randomicamente una cassa in cui accodarsi, aspetta che venga servito e poi lascia il supermercato. Se la cassa in cui si sceglie di far la fila dovesse essere chiusa, il cliente viene notificato e procede con la scelta di una nuova coda tra quelle disponibili. Se non dovesse effettuare acquisti, chiede prima di terminare il permesso al thread “controllo_clienti_P0”. In entrambi i casi, prima della fine effettiva del thread, tutti i dati acquisiti vengono salvati nel file “resoconto.log”.

2. Implementazione: scelte e dettagli

Tutte le strutture create ed utilizzate, così come le funzioni e le procedure, sono inserite e descritte brevemente nel file “varie.h” che verrà incluso in supermercato.c. Per le code sono state create delle linked list attraverso una concatenazione di struct per permettere loro di assumere la politica FIFO. Sono state realizzate anche delle funzioni apposite in modo da crearle, gestirle e distruggerle.

La comunicazione fra thread è stata implementata con l’utilizzo di variabili globali, il cui accesso in mutua esclusione viene garantito attraverso le mutex e le condition variables.

Per acquisire i dati dal file “conf.txt” è stata creata una funzione “inizializza” appositamente scritta per permette di leggere il file di configurazione rispettando la sua specifica “formattazione”.

Per l’id di casse e clienti, è stato scelto di utilizzare invece di un intero, l’id stesso del thread.

In caso di chiusura di una cassa, il cassiere deposita i dati acquisiti in una lista. Nel caso dovesse poi essere riaperta la cassa, a gestirla sarebbe un cassiere diverso con un id diverso. Per questo motivo, nel caso il numero massimo consentito di casse sia K, negli output finali potrebbero esserci K+X casse a mostrare i propri dati, dove X sono le casse aperte oltre K. Questo perché è stato scelto di mostrare i dati collezionati da ogni “cassiere” e non quelli collezionati “a tale cassa” . Nonostante ciò, è garantito K come valore massimo di casse attive contemporaneamente.

I segnali di SIGQUIT e SIGHUP sono stati gestiti attraverso l'installazione di un handler chiamato “gestore_segnali” che setta le variabili globali “sigquit” e “sighup” ad 1 non appena arriva il rispettivo segnale. In caso di SIGQUIT è stata una scelta personale quella di sostituire i dati acquisiti dal cliente mostrati negli output, con la dicitura “Interrotto da SIGQUIT”.