Simulazione di un sistema multithreaded e multiprocesso di un supermercato

Progetto di Laboratorio di Sistemi Operativi

a.a 2019/20

**Ferraro Domenico**

Matricola 559813

Corso B

Sommario

[1 Introduzione 1](#_Toc49090137)

[2 Supermercato 1](#_Toc49090138)

[Algoritmo di cambio cassa 2](#_Toc49090139)

[Clienti 2](#_Toc49090140)

[Cassieri 2](#_Toc49090141)

# Introduzione

La simulazione del supermercato è realizzata mediante un sistema multithreaded e multiprocesso. Le entità principali sono il direttore ed il supermercato e sono realizzate con due processi differenti. Il processo Direttore si occupa dell’apertura del supermercato e quindi lancia il processo Supermercato e la comunicazione tra i due processi avviene mediante socket AF\_UNIX. Il processo Supermercato modella i cassieri e le loro code, i clienti e le loro uscite ed entrate a gruppi mediante più thread POSIX.

# Supermercato

Come il nome lascia intendere, questo processo modella l’intero supermercato quindi tutti i clienti, la gestione dell’accesso a gruppi, tutti i cassieri e la gestione delle loro code. Lo fa mediante i seguenti thread POSIX: thread principale, thread gestore dei segnali, due thread per ogni cassiere e un pool di thread grande quanto il massimo numero di clienti che può entrare nel supermercato.

Il thread principale mediante multiplexing si occupa della gestione delle richieste via socket AF\_UNIX da parte del processo Direttore e della comunicazione via pipe con il thread signal handler. Il thread signal handler cattura il segnale ricevuto dal processo Direttore e lo rimanda via pipe al thread principale il quale lo traduce nel relativo stato di chiusura del supermercato. Il thread principale riceve tramite socket dal direttore le richieste di apertura o chiusura di una cassa e se è concesso o meno ad un determinato cliente di uscire dal supermercato. In tali circostanze, il thread principale traduce in maniera concreta le volontà espresse dal direttore, quindi comunica al thread cassiere di aprire/chiudere la cassa e comunica al relativo thread cliente che gli è concesso uscire o meno.

Ogni thread è in grado di riconoscere da solo se il supermercato sta chiudendo o meno e svolge autonomamente la sua terminazione. Il thread principale, quindi, si occupa solo di cambiare lo stato del supermercato in base al segnale ricevuto dal processo Direttore e di svegliare eventuali thread dormienti su condition variables.

## Algoritmo di cambio cassa

Quando una cassa è aperta, i thread clienti si mettono in coda e attendono di essere serviti. Ogni cliente, quando è in coda, si chiede in maniera periodica se la cassa in cui si trova è la più conveniente o meno e quale delle altre casse è la migliore in quel determinato istante. Ad ogni cassa è associata una coda e per via del tempo di servizio di un cassiere, ogni coda potrebbe essere più lenta delle altre. Inoltre, anche il numero di clienti in coda e quanti prodotti ogni cliente possiede influisce su quanto una coda sia più o meno veloce.

## Clienti

Mentre un cliente è in coda, diversi eventi possono avvenire: la cassa viene chiusa, il supermercato viene chiuso. Inoltre, il cliente vuole valutare se è più conveniente accodarsi in un’altra cassa aperta (*algoritmo di cambio cassa*). Per gestire queste situazioni, un cliente in coda non rimane totalmente dormiente aspettando di essere servito, ma si risveglia in maniera periodica. In questo modo è in grado di reagire in maniera opportuna ai suddetti eventi.

## Cassieri

Ogni cassiere è implementato con due thread: il thread cassiere che svolge il ciclo di vita del cassiere ed il thread notificatore che, quando la cassa è aperta, comunica via socket AF\_UNIX al direttore ad intervalli regolari il numero di clienti in coda.