Progetto di Laboratorio di Sistemi Operativi a.a. 2019-20

Nome: Abramo
Cognome: Maffei
N.Matricola: 552887
Corso: A

ELEMENTI DEL SISTEMA:
1) Main
2) Direttore
3) Gestore Clienti
4) Cliente
5) Gestore Casse
6) Cassa

1) MAIN
Si occupa di avviare i Thread "Gestore Clienti", "Gestore Casse", " Direttore1", "Direttore2".
Si mette in attesa del Thread "Amministratore1" ed una volta terminato avvia una funzione di pulizia della memoria dinamica. Alla fine termina il processo.

2)DIRETTORE
Esistono due tipi di Direttore:
- Direttore2: E' un Thread che attente la terminazione dei processi "Gestore Clienti" e "Gestore Casse" quando terminano manda un messaggio al "Direttore1" di terminare.
- Direttore1: E' un Thread che attende istruzioni le quali possono essere o di "Uscita Senza Prodotti" da un Cliente o di "Aggiornamento" da una cassa. Nel caso di "uscita senza prodotti" invia un messaggio al Cliente dandogli il permesso di uscire, mentre nel caso "Aggiornamento" aggiorna il proprio stato interno e se necessario prende una decisione di Aprire/Chiudere una cassa. Quando il "Direttore2" invia il messaggio di chiusura, il "Direttore1" termina la propria esecuzione.

3)GESTORE CLIENTI
E' un processo che si occupa di mantenere/gestire i clienti, infatti è esso stesso che crea i Thread Clienti e ne fa entrare altri nel momento in cui il numero di clienti attivi scende sotto una determinata soglia.
Inoltre è colui che aspetta la terminazione di tutti i Thread "Cliente" e dopo termina, indicando la conclusione di tutti i Thread Cliente.

4)CLIENTE
Ogni Cliente genera una propria Struct in cui salva le proprie caratteristiche, una volta creato esso si mette in attesa per un tempo casuale simulando la ricerca dei prodotti.
Una volta finita l'attesa ha diverse possibilità:
- 1. Se non ha prodotti manda la richiesta di uscita al Direttore e quando riceverà un messaggio di accettazione uscirà.
- 2. Se ha prodotti si metterà in attesa su una coda scelta casualmente da quelle aperte.
- 3. Se è in corso una Evacuazione(SIGQUIT) o una Chiusura(SIGHUP) esce direttamente senza pagare e senza prodotti.
- 4. Se era in coda e in seguito viene chiusa la cassa si mette in coda in un'altra coda.

5)GESTORE CASSE
Si occupa di generare le prime casse ed inizializzare le code, dopo di che aspetta la fine di tutte le casse per poi terminare, identificando la conclusione di tutti i Thread Cassa.

6)CASSA
Ogni cassa genera una propria Struct avente le proprie caratteristiche, una volta creata si mette in ascolto su una coda specifica prendendo/aspettando Clienti.
La Cassa ha diverse possibilità:
- 1. Arriva un cliente, aspetta un tempo determinato da caratteristiche proprie e dal numero di prodotti acquistati ed in fine invia un messaggio al Thread Cliente di avvenuto pagamento.
- 2. Ogni quantità di tempo invia un aggiornamento al Thread Direttore1 in merito al numero di elementi nella sua coda.
- 3. Se il supermercato sta Chiudendo(SIGHUP) non accetta ulteriori Clienti nella coda, finisce di far pagare i presenti e termina.
- 4. Se il supermercato sta Evacuando(SIGQUIT) libera immediatamente tutti i Clienti dalla coda e termina.
- 5. Se la coda deve chiudere su decisione del Direttore, libera i Clienti in coda e termina.

COMUNICAZIONI:
Esistono 5 tipi di comunicazioni:
1. Cliente -> Direttore, eseguita con una coda di messaggi FIFO unica.
2. Direttore -> Cliente, eseguita con una coda di messaggi FIFO una per ogni Cliente.
3. Cassa -> Cliente, eseguita con una coda di messaggi FIFO una per ogni Cliente.
4. Cassa -> Direttore, eseguita con una coda di messaggi FIFO unica.
5. Direttore -> Cassa, eseguita con una coda di messaggi FIFO una per ogni Cassa.
La scelta di utilizzare delle code FIFO è data dalla semplicità di implementazione e dalla corrispondenza logica tra le comunicazioni effettive in un centro commerciale e quelle nel programma.
Inoltre era richiesta da specifiche del progetto una coda FIFO per le Code.

LE CODE:
Sono una struttura dati persistente, ovvero non vengono cancellate o create in base ai Thread delle casse, ma bensì sono inizializzate con l'avvio del "Gestore Casse" e cancellate alla terminazione di esso.
In questo modo all'apertura di una cassa gli viene affidato un ID (0<=ID<K) di una coda, alla quale la Cassa farà riferimento, pertanto si possono aprire/chiudere infinite volte i Thread Cassa senza avere problemi di interazioni con le code.
Inoltre si facilità il mantenimento dei dati di esecuzione es."numero clienti".

RANDOM:
Esistono due varabili da generare random, uno riferito alle caratteristiche di ciascun Cliente e uno per le caratteristiche di ciascuna Cassa.
La funzione random viene impostate su un seed generato dall'unione di TIME(NULL) e TID del Thread, in questo modo non avvengono conformità di generazione, in quanto usando solo TIME, avremmo avuto le solite caratteristiche per i Clienti che accedono insieme.

TEMPORALIZZAZIONE:
Nell'esecuzione del programma su alcuni Thread sono stati aggiunte delle attese di 1 secondo.
Per esempio nella cassa, nel caso in cui la sua coda fosse vuota, eseguirebbe un ciclo infinito nel quale il controllo sulle variabili di chiusura/evacuazione sarebbero costantemente in mutua esclusione, non permettendo agli altri Thread di utilizzarle e quindi bloccando il sistema. Per questo è stato aggiunto nel caso di "coda vuota" 1 secondo di delay per non ostruire il sistema.
Per lo stesso motivo sia all'interno del "Gestore Clienti" sia nel "Gestore Casse" è stato aggiunto un delay di 1s all'interno del proprio ciclo per non consentirgli la mutua esclusione continua su delle variabili.

STRUTTURA DEI FILE:
./supermercato.c - Main del programma.
./supermercato.h - Funzioni e strutture dati condivise.
./cliente.c - Tutte le funzioni necessarie al funzionamento dei Clienti.
./cassa.c - Tutte le funzioni necessarie al funzionamento delle Casse.
./boss.c - Tutte le funzioni necessarie al funzionamento dei Direttori.
./code.c - Tutte le funzioni per le interazioni con le Code.
./messaggi.c - Tutte le funzioni per l'invio/ricezione dei messaggi.
./signal.c - Handlers dei segnali.
./log.c - Tutte le funzionalità di scrittura su file.
./Makefile - Automatizza processi di compilazione/avvio.
./valgrind.sh - Script bash per l'avvio del programma.
./analisi.sh - Script bash per l'avvio dell'analisi dei log.
./clean.sh - Script bash per rimozione dei file .o.
./test/test1.txt - Costanti configurabili all'avvio del programma.
*./data/valgrind.tx*t - Risultato dell'esecuzione con Valgrind.
./data/clientiOut.txt - Logs di uscita dei clienti.
*./data/casseOut.tx*t - Logs di uscita delle casse.

COMANDI:
make clean – Cancella gli eseguibili.
make – Compila il programma.
make test – Avvia il programma con il relativo ./data/test1.txt
make analisi – Avvia lo script di analisi.
make help – Lista comandi possibili.