Autore: Fabrizio Cau

Matricola: 508700

Corso A Informatica

Relazione del progetto Laboratorio di Sistemi Operativi a.a 2019-20

Il progetto riguarda una simulazione delle operazioni di base di un supermercato. Questo è stato realizzato attraverso la creazione di due processi in esecuzione parallela che rappresentano il direttore e il supermercato. I due processi inizialmente raccolgono i parametri di configurazione in tre modalità sequenziali: prima vengono impostati i parametri di default, successivamente viene letto il file di configurazione config.txt ed infine i parametri inseriti al lancio e passati al main. Questi ultimi due possono sovrascrivere alcuni o tutti i dati di default, l'ultimo ha naturalmente la priorità sui precedenti. A questo punto viene impostata e stabilita una connessione di tipo socket AF_UNIX in cui il supermercato funziona da server e il direttore da client.

Processo direttore. Il processo direttore se lanciato col parametro FORK_FROM_MNG=1 lancerà il processo supermercato mediante una fork seguita da una exec, in ogni caso successivamente si metterà in attesa della connessione socket. Tramite scambio di messaggi via socket attenderà alcuni valori iniziali quali, il pid del processo supermercato e un messaggio "ready" che conferma che sono state inizializzate le principali strutture e thread del supermercato. A questo punto invia i messaggi di apertura delle prime casse che terminerà con un messaggio di "start" e inizierà il ciclo di lettura delle notifiche dal supermercato e a ciascuna l'eventuale risposta con l'azione da prendere. Il processo terminerà "l'ascolto" dei messaggi di gestione alla ricezione di un ulteriore particolare messaggio ("store-lastmsg") e si metterà in attesa del termine del processo supermercato che sarà preceduto dal messaggio "store-end".

Socket. I messaggi scambiati tra i due processi, seguono delle regole stabilite: ad esempio se il direttore riceve un messaggio che inizia per "ck", è una notifica inviata da una cassa, e dunque sarà seguito dal numero della cassa e il numero di clienti in coda; quando inizia per "cu" si tratta di un cliente con 0 prodotti che richiede il permesso di uscire. Nel primo caso il direttore in base ad un breve algoritmo di decisione può chiudere la cassa che ha notificato o aprirne una a caso. Nel secondo caso invece invia un messaggio per confermare l'uscita del cliente. I messaggi vengono gestiti in modo FIFO, dunque i primi che arrivano sono i primi ad essere gestiti e ai quali vi è una risposta (eventualmente), gli ultimi verranno messi in coda.

Processo supermercato. Il processo supermercato dopo aver stabilito la connessione socket lancia un thread che si occupa di inizializzare la struttura dati delle casse, del lancio di un thread per ogni cassa e si interfaccia con il processo direttore, ovvero della ricezione e interpretazione dei messaggi ricevuti. Una volta inizializzate le casse viene lanciato un thread che inizializza la struttura dati dei clienti, lanciando anche per questi un thread per ognuno, infine si occupa della generazione costante di "nuovi clienti in attesa di entrare" tramite una lista di E interi (contenente gli id dei futuri clienti) gestita in mutua esclusione.

Thread Clienti. Ogni cliente è gestito da un thread, ma alla sua uscita il thread non termina, viene infatti "riciclato" mettendosi in attesa di nuovi elementi forniti dal generatore di clienti (genCustomer). Ogni cliente ha nella sua struttura dati una variabile di stato che indica se il thread è vuoto, se sta gestendo un cliente e in questo caso se il cliente è in cassa o se viene servito. Questa variabile rende possibile la comunicazione tra il thread cliente e un thread di supporto da esso generato che si occupa di verificare ed effettuare ad intervalli stabiliti un eventuale cambio di cassa. Le operazioni di modifica e lettura dei dati del cliente, soprattutto nel caso della variabile di stato sono fatte in mutua esclusione. Tutte le attese sulla variabile di stato sono di tipo passivo.

Thread Casse. Ogni cassa è gestita da un thread che attende passivamente l'arrivo di clienti in coda, quando viene risvegliato controlla la coda ed eventualmente effettua la pop. Ogni cassa ha un thread di supporto che si occupa di notificare ad intervalli stabiliti il numero di clienti in coda al processo direttore attraverso il canale socket.

Librerie. Sono state realizzate due librerie: una, chiamata datastruct, utilizzata dal processo supermercato si occupa della gestione della struttura dati di clienti e casse e comprende l'inizializzazione, le funzioni di push e pop dalle varie code, apertura e chiusura delle casse e diverse funzioni su supporto relative, comunque, alle strutture dati principali dichiarate nel suo header file. La seconda libreria, mylib, fornisce funzioni di supporto ad entrambi i processi non relative alla struttura dati principale, e macro di controllo dei valori di ritorno.

Terminazione. La terminazione dei processi inizia tramite l'invio di uno tra i due segnali SIGHUP e SIGQUIT inviato dal makefile al processo direttore. Questo invierà a sua volta lo stesso segnale al processo supermercato che farà avanzare il flusso di un thread (in attesa su una sigwait) realizzato appositamente per la chiusura (quit). Questo in base al segnale ricevuto imposterà, con una determinata sequenza, le variabili di stato del supermercato usate in condivisione tra i vari thread e, segnalando l'uscita dalle attese su variabili di condizione, attenderà l'uscita di ciascun thread. Al termine di tutti stamperà i dati statistici sul file di log e libererà tutta la memoria allocata dai vari thread.

NB. Per garantire una certa robustezza del codice e sicurezza dell'esecuzione di tutte le istruzioni, mediante delle macro, vengono controllati i valori di ritorno di quasi tutte le funzioni di libreria fondamentali per la corretta esecuzione dei processi e delle chiamate di sistema. Le macro alcune system call, in particolare quelle di connessione, lettura e scrittura sul canale socket, implementano un ciclo che permette di riprendere l'istruzione interrotta da eventuali segnali POSIX.