Relazione Progetto Laboratorio di Sistemi Operativi

Gabriele Masciotti 578318

L'archivio contiene tutto il necessario per poter eseguire il programma oggetto del progetto. Il contenuto è costituito dal file *supermercato.c* che contiene il codice sorgente che sviluppa la consegna, il *Makefile*, un header file (*utilities.h*) che contiene funzioni ausiliarie utilizzate nel programma e lo script *analisi.sh* per il parsing delle statistiche d'esecuzione nel terminale.
Perché il programma funzioni correttamente è necessario che suddetti file rimangano sempre nello stesso folder.

Makefile

Per generare l'eseguibile del programma è sufficiente eseguire il comando 'make'.
Per eseguire il programma è necessario un file di configurazione. Per effettuare una simulazione con dati a propria scelta si può eseguire il comando 'make conf' il quale genera un file di configurazione vuoto che è possibile editare aggiungendo i valori desiderati. Altrimenti si può effettuare la simulazione predefinita (come da specifica) digitando il comando 'make test'; questo comando genera un file di configurazione con i valori richiesti dalla consegna ed avvia la simulazione, riportando, al termine, il sunto delle statistiche collezionate. Queste ultime sono stampate a video sul terminale dallo script **analisi.sh** che legge il file di log che è stato modificato più di recente. Questo consente all'utente di mantenere più file di log (modificandone il nome, altrimenti verrebbero sovrascritti) in modo da poter confrontare diverse istanze d'esecuzione.
Il comando 'make clean' permette di ripulire la cartella di lavoro dall'eseguibile e dai file di log generati.

supermercato.c

Il file supermercato.c contiene il codice sorgente per il funzionamento del programma.
All'inizio del file sono presenti tutte le strutture dati necessarie e un insieme di risorse condivise protette dai relativi mutex.
Le simulazioni cominciano con una **fase di inizializzazione**; in questa fase: viene letto il file di configurazione presente nella directory che deve chiamarsi "config.txt", sono impostati i valori di funzionamento, vengono preparate tutte le casse del supermercato, il buffer di comunicazione dei cassieri con il direttore, la gestione dei segnali di chiusura e viene aperto il file di log.
A questo punto viene **aperto il supermercato**; si aprono le prime "I" casse generando i primi "I" thread cassieri, si generano i primi "C" thread clienti e si genera il thread direttore.
Nel **corso della simulazione** il thread main si occupa di generare i clienti via via che quelli che sono stati serviti escono dal supermercato (terminano). I thread *cassieri* generano un altro thread addetto alla comunicazione periodica con il direttore; prima di cominciare a servire ogni cliente che si trova in coda alla propria cassa, il cassiere controlla che il direttore non abbia chiuso la cassa o che il supermercato non sia stato chiuso. I thread *clienti* entrano nel supermercato (vengono generati) e spendono un determinato periodo di tempo per gli acquisti; in particolare i thread attendono tale periodo di tempo sospendendosi temporaneamente su una variabile di condizione comune, in modo tale che se il supermercato dovesse essere chiuso con SIGQUIT mentre stanno ancora facendo acquisti, risvegliatisi dalla broadcast del thread main si accorgerebbero che devono uscire immediatamente (terminare). Trascorso questo tempo se il cliente non ha effettuato acquisti (prodotti=0) si mette in coda per attendere il permesso del direttore di uscire dal supermercato. Altrimenti sceglie una cassa random tra quelle aperte, si inserisce in fondo alla coda e attende di essere servito dal cassiere. Se il cliente che si risveglia dall'attesa non è stato servito, significa che la cassa è stata chiusa mentre aspettava. Se il supermercato è ancora aperto, semplicemente cambia coda; se invece la cassa è stata chiusa perché il supermercato è anch'esso stato chiuso con segnale SIGQUIT allora il cliente esce immediatamente. I clienti usciti dopo un SIGQUIT inseriscono il proprio tid in una lista apposita in modo da permettere al main di fare le join ed evitare memory leak. Il thread *direttore* si occupa di decidere se aprire o chiudere le casse in base alle informazioni ricavate dal buffer di comunicazione riempito dai sotto-thread di comunicazione dei cassieri. Concede, appunto, anche il permesso ai clienti che non hanno effettuato acquisti di uscire dal supermercato.
In **fase di chiusura**; se è stato ricevuto il segnale SIGQUIT il thread main indica a tutti gli altri thread di terminare il prima possibile, attende poi che tutti siano terminati prima di liberare le risorse in uso e di terminare a sua volta. Se è stato ricevuto il segnale SIGHUP il thread main smette di generare nuovi clienti e lascia che gli altri thread continuino la loro esecuzione normale fin quando tutti i clienti presenti non sono

usciti, quindi tutte le casse chiuse e il direttore terminato. Dopo aver atteso il direttore, libera le risorse e termina a sua volta.
Nel corso delle simulazioni, prima di terminare, i clienti scrivono sul **file di log** le informazioni necessarie per le statistiche d'esecuzione da raccogliere. I cassieri contano i clienti serviti, i prodotti venduti e i tempi d'apertura e di servizio. Prima di terminare il thread main scrive nel file di log le statistiche di ogni cassa e poi chiude il file.
Ulteriori dettagli sono forniti nei numerosi commenti a fianco al codice.

analisi.sh

L'ultimo file di log modificato (o creato) viene letto dallo script bash analisi.sh il quale riporta nel terminale (sullo standard output) tutte le righe del file.