Progetto SO 2019/2020 di Federico Giannotti – versione semplificata

Il documento ha lo scopo di descrivere e motivare le scelte sull' implementazione del progetto. Per fare ciò, il documento si divide in più punti contenenti gli aspetti ritenuti cruciali. Le informazioni già contenute nel testo del progetto non verranno ulteriormente trattate, in modo da contenere solo i dettagli relativi alle scelte personali.

-Variabili globali

Tutti i valori contenuti nel file di configurazione sono assegnati a delle variabili globali all'interno del processo. Oltre alle variabili legate ai valori di configurazione, sono presenti altre variabili alle quali viene dato un valore di default ad ogni esecuzione del programma. La scelta dell'utilizzo di variabili globali è legata all'esecuzione del progetto. Dato che solamente un processo è in esecuzione, è stato ritenuto accettabile utilizzare variabili globali, così da poter evitare un ripetuto passaggio di parametri ai vari threads e permettere ad essi di dialogare usandole come una variabile condivisa.

-Strutture dati

Le strutture dati all'interno del progetto sono due: cliente e cassa.

Cassa contiene oltre agli attributi standard richiesti nel testo del progetto, una variabile condizione, un puntatore ad una struttura dati di tipo cassa ed uno ad una struttura dati di tipo cliente. La variabile di condizione serve a sospendere la cassa quando si trova in attesa di un cliente. Il puntatore alla cassa successiva è utilizzato per accedere facilmente a tutte le casse, mentre il secondo puntatore descritto è utilizzato per accedere facilmente a tutti i clienti che stanno facendo la fila in quella determinata cassa. Cliente contiene oltre agli attributi standard richiesti nel testo del progetto, un puntatore alla successiva struttura dati cliente. Questo permette di simulare una fila di clienti in coda ad una cassa.

-Threads: cliente, cassa e direttore

Il thread cliente svolge un compito molto semplice: crea una struttura dati cliente, esegue la funzione nanosleep (che simula l'acquisto dei prodotti all'interno del supermercato), seleziona una cassa e termina. Il thread termina immediatamente perché non c'è più bisogno che esegua ulteriori azioni. L'oggetto cliente che ha creato viene poi puntato dalla struttura dati delle casse (nella funzione di selezione cassa) e gestito dai threads cassa. È stato possibile adottare questa soluzione dato che nella specifica del progetto semplificato i clienti non potevano cambiare fila autonomamente. In quel caso il thread cliente sarebbe dovuto rimanere in esecuzione per permettere tale funzionalità. Ad ogni cliente corrisponde un nuovo thread, ma il fatto che terminino in poco tempo permette un utilizzo molto basso delle risorse durante l'esecuzione del processo.

Il thread cassa crea una struttura dati cassa che viene successivamente inserita in una lista circolare contenente tutte le casse. Il thread cassa può trovarsi in due stati, aperto o chiuso. I due valori sono rappresentati con un attributo nella struttura dati. Il thread rimane in attesa passiva se è chiuso o se la fila dei clienti è vuota. Tutti i threads cassa sono spawnati all'inizio del processo, anche se alcuni di loro partiranno nello stato di chiuso. Questa soluzione permette di avere una struttura ben definita fin dall'inizio. Il thread cassa genera un thread secondario support il quale ha lo scopo di controllare in un intervallo di tempo definito il numero di clienti presenti in coda e successivamente aggiornare l'attributo corrispondente nella struttura dati cassa. Quando il thread cassa è in attesa passiva, lo è anche il thread support dato che condividono la solita variabile condizione.

Il thread direttore non crea nessuna scrittura dati ma esegue un ciclo while all'interno del quale esegue due funzioni: analizza le code delle casse (eventualmente apre o chiude una di esse) e analizza la coda composta dai clienti senza prodotti, dando loro il permesso di uscire. Quando la variabile globale CLOSE cambia

valore il thread direttore termina. Dato che nella versione semplificata non è un processo separato da quello del supermercato, il thread direttore termina naturalmente prima del processo supermercato.

-Processo supermercato

In questa sezione viene illustrato lo svolgimento del programma in maniera riassuntiva.

1. Assegnamento parametri iniziali.
2. Creazione threads cassa, direttore e cliente.
3. I threads svolgono le mansioni precedentemente descritte.
4. Il processo supermercato entra in un ciclo while: fino a che non si verificano le condizioni di interruzione, continuerà a spawnare nuovi clienti se le condizioni lo permetteranno.
5. Quando il processo supermercato riceve il segnale di interruzione si mette in attesa passiva aspettando che le casse processino i clienti rimanenti.
6. SIGHUP: i clienti rimasti all'interno del supermercato pagano la loro spesa. Finiti i clienti, i threads casse terminano. SIGQUIT: Una funzione speciale libera i clienti all'interno del supermercato. Alle casse sembrerà di non aver più clienti da processare, terminando immediatamente.
7. Il processo supermercato esegue la join su tutti i threads che si sono creati durante l'esecuzione del programma ed esegue la free sulla memoria allocata.

-Makefile

Il progetto è stato realizzato con tre file sorgenti e un header. Il Makefile genera i file oggetto e l'eseguibile del programma. Non viene creata nessuna libreria. La motivazione di questa scelta risiede nel fatto che i tre file sorgenti sono strettamente dipendenti l'uno dall'altro e non è possibile riutilizzare uno di essi indipendentemente dagli altri. In effetti si sarebbe potuto realizzare un unico file contenente le tutte le funzioni e il main, ma per praticità è stato diviso in tre parti principali. Il Makefile non contiene il target all dato che il processo direttore non è presente. Contiene i target clean, cleanall e test. Il primo rimuove solamente l'eseguibile, mentre il secondo anche i file oggetto e il file di logs. Il target test avvia l'eseguibile insieme ad uno script, il quale dopo un intervallo di tempo prestabilito, invia un segnale al processo. Conclusa l'esecuzione del programma, un ultimo script stampa a schermo il contenuto del file di logs.

-File di configurazione

Per settare i parametri iniziali si può modificare il file di configurazione contenuto nella directory conf. Dato che è strettamente necessario mantenere la struttura del file di configurazione per l'esecuzione del programma, non è stata ritenuta una buona scelta permettere di passare un proprio file di configurazione.

-Note

Per testare il programma con valgrind è presente una parte di codice commentata. Basta trasformare il commento in codice ed usare valgrind direttamente sull'eseguibile. In questo caso il programma terminerà dopo una serie di cicli di creazione clienti (è possibile impostare a piacere il numero di tali cicli).

Per simulare il tempo di acquisto e pagamento è stata utilizzata la funzione nanosleep. I tempi di attesa sono stati trasformati da nanosecondi a millisecondi come richiesto dalla specifica del progetto.