Università di Pisa-- Dipartimento di Informatica
Corso di Laurea in Informatica

Progetto di Laboratorio di Sistemi Operativi

a.a. 2019-20
Docenti: G. Prencipe (Corso A), M. Torquati (Corso B)
Data pubblicazione 2 Maggio 2020
Revisione: 5 Maggio 2020

Introduzione

Lo studente dovrà realizzare la simulazione di un sistema che modella un supermercato con *K* casse e frequentato da un certo numero di clienti. Il supermercato è diretto da un direttore che ha facoltà di aprire e chiudere una o più casse delle *K* totali (lasciandone attiva almeno una). Il numero dei clienti nel supermercato è contingentato: non ci possono essere più di *C* clienti che fanno acquisti (o che sono in coda alle casse) in ogni istante. All'inizio, tutti i clienti entrano contemporaneamente nel supermercato, successivamente, non appena il numero dei clienti scende a *C-E* ($0<E<C$), ne vengono fatti entrare altri *E*. Ogni cliente spende un tempo variabile *T* all'interno del supermercato per fare acquisti, quindi si mette in fila in una delle casse che sono in quel momento aperte ed aspetta il suo turno per "pagare" la merce acquistata. Periodicamente, ogni cliente in coda, controlla se gli conviene cambiare coda per diminuire il suo tempo di attesa. Infine esce dal supermercato.
Un cliente acquista fino a *P>=0* prodotti. Ogni cassa attiva ha un cassiere che serve i clienti in ordine FIFO con un certo tempo di servizio. Il tempo di servizio del cassiere ha una parte costante (diversa per ogni cassiere) più una parte variabile che dipende linearmente dal numero di prodotti acquistati dal cliente che sta servendo.
I clienti che non hanno acquistato prodotti (*P=0*), non si mettono in coda alle casse, ma prima di uscire dal supermercato devono attendere il permesso di uscire dal direttore.

Il supermercato

Il supermercato è modellato come un singolo processo multi-threaded con al più *K* thread attivi come cassieri e *C* thread attivi corrispondenti ai clienti che sono nel supermercato. Eventuali altri thread di "supporto" (oltre al thread main) possono essere presenti nel sistema a discrezione dello studente.
Il direttore è un processo separato dal processo supermercato. Quando necessario, i due processi interagiscono tramite socket di tipo AF_UNIX e segnali POSIX. Al loro avvio, entrambi i processi leggono i rispettivi parametri di configurazione da un file comune chiamato *config.txt* (non è un header file). Il formato di tale file non viene specificato e deve essere deciso dallo studente.
Il supermercato all'inizio apre con un numero limitato di casse (ad esempio 1 sola). Tale valore iniziale è definito nel file di configurazione.
Il supermercato chiude quando il direttore riceve un segnale SIGQUIT o SIGHUP. Nel primo caso la chiusura deve essere immediata, ossia gli eventuali clienti ancora presenti nel supermercato non vengono serviti ma vengono fatti uscire immediatamente. Nel secondo caso (SIGHUP), non vengono più fatti entrare nuovi client ed il supermercato termina quando tutti i clienti nel supermercato escono perchè hanno terminato gli acquisti.
Prima di terminare, il processo supermercato scrive in un file di log (il cui nome è specificato nel file di configurazione) tutte le statistiche collezionate dal sistema durante il periodo di apertura. Tra queste: il numero di clienti serviti, il numero di prodotti acquistati; per ogni cliente: il tempo di permanenza nel supermercato, il tempo di attesa in coda, se e quante volte ha cambiato coda, il numero di prodotti acquistati; per le casse: il numero di clienti serviti, il tempo di ogni periodo di apertura della cassa, il numero di chiusure, il tempo di servizio di ogni cliente servito.

Il cassiere

I clienti si accodano in modo random alle casse aperte nel momento in cui hanno terminano gli acquisti, e vengono serviti in ordine di arrivo dal cassiere. Il thread cassiere impiega un tempo fisso (diverso per ogni cassiere nel range *20-80* millisecondi) ed un tempo variabile che dipende in modo lineare dal numero di prodotti che il cliente ha acquistato (il tempo di gestione di un singolo prodotto da parte di un cassiere è fisso, ed è

specificato nel file di configurazione). Il direttore viene informato ad intervalli regolari dai cassieri (l'ampiezza di tale intervallo è definita anch'essa nel file di configurazione) sul numero di clienti in coda alla cassa. Quando una cassa viene chiusa dal direttore, il thread cassiere termina dopo aver servito il cliente corrente. Gli eventuali altri clienti in coda si devono rimettere in coda in altre casse.

Il cliente

Ogni cliente che entra nel supermercato ha associato un tempo per gli acquisti che varia in modo casuale da *10* a *T>10* millisecondi, ed un numero di prodotti che acquisterà che varia da 0 a *P>0*. Tali valori vengono associati al thread cliente all'atto della sua entrata nel supermercato.
Il cliente in coda ad una cassa può decidere di spostarsi in un'altra cassa. La decisione viene presa in base al numero di clienti che ha davanti a lui ed al numero di clienti in coda nelle altre casse. In particolare, ogni *S* millisecondi (*S*, *T* e *P* sono parametri specificati nel file di configurazione), ogni cliente decide se spostarsi o meno. L'algoritmo che implementa la decisione di spostarsi è lasciato allo studente. Il cliente che decide di spostarsi, si prende il rischio che la cassa in cui si muoverà potrebbe essere chiusa (nel frattempo) dal direttore. In tal caso perderà la posizione che aveva nella vecchia cassa e dovrà rimettersi in coda in una delle casse aperte.
Un cliente con 0 prodotti acquistati non si mette in coda in nessuna cassa, ma dovrà informare il direttore che intende uscire e dovrà attendere la sua autorizzazione prima di farlo.

Il direttore

Il direttore, sulla base delle informazioni ricevute dai cassieri, decide se aprire o chiudere casse (al massimo le casse aperte sono *K*, ed almeno 1 cassa deve rimanere aperta). La decisione viene presa sulla base di alcuni valori soglia *S1* ed *S2* definiti dallo studente nel file di configurazione. *S1* stabilisce la soglia per la chiusura di una cassa, nello specifico, definisce il numero di casse con al più un cliente in coda (es. *S1=2:* chiude una cassa se ci sono almeno 2 casse che hanno al più un cliente). *S2* stabilisce la soglia di apertura di una cassa, nello specifico, definisce il numero di clienti in coda in almeno una cassa (es. *S2=10:* apre una cassa (se possibile) se c'è almeno una cassa con almeno 10 clienti in coda).
Quando il processo direttore riceve un signale SIGQUIT o SIGHUP informa immediatamente il processo supermercato inviandogli lo stesso segnale. Nel caso di SIGHUP, il supermercato non deve più far entrare altri clienti e deve attendere che tutti i clienti presenti nel supermercato terminino gli acquisti. Nel caso di segnale SIGQUIT, il supermercato chiude immediatamente facendo uscire tutti i clienti al suo interno. Il direttore attende che il supermercato termini prima di terminare a sua volta.

Makefile

Il progetto dovrà includere un Makefile avente, tra gli altri, i target *all* (per generare gli eseguibili del programma supermercato e direttore), *clean* (per ripulire la directory di lavoro dai file generati, socket file, logs, librerie, etc.), e due target di tests: *test1* e *test2*. Il target *test1* deve far partire il processo direttore, quindi il processo supermercato con i seguenti parametri di configurazioni (quelli non specificati, sono a scelta dello studente): *K=2, C=20, E=5, T=500, P=80, S=30* (gli altri parametri a scelta dello studente). Il processo supermercato deve essere eseguito con valgrind utilizzando il seguente comando: "valgrind –leak-check=full". Dopo 15s deve inviare un segnale SIGQUIT al processo direttore. Il test si intende superato se valgrind non riporta errori né memoria non deallocata (il n. di malloc deve essere uguale al n. di free).
Il *test2* deve lanciare il proccesso direttore che provvederà ad aprire il supermercato lanciando il processo supermercato con i seguenti parametri (gli altri a scelta dello studente): *K=6, C=50, E=3, T=200, P=100, S=20*. Dopo 25s viene inviato un segnale SIGHUP, quindi viene lanciato lo script Bash di analisi (*analisi.sh*) che, al termine dell'esecuzione del supermercato, fornirà sullo standard output un sunto delle statistiche relative all'intera simulazione appena conclusa. Il test si intende superato se non si producoro errori a run-time e se il sunto delle statistiche relative alla simulazione riporta "valori ragionevoli" (cioè, non ci sono valori negativi, valori troppo alti, campi vuoti, etc...).

Lo script *analisi.sh*

Lo studente dovrà realizzare uno script Bash con nome *analisi.sh* che effettua il parsing del file di log prodotto dal processo supermercato al termine della sua esecuzione, e produce un sunto della simulazione. Nello specifico, lo script produce sullo standard output le seguenti informazioni.

Per i clienti:
| ***id cliente*** | ***n. prodotti acquistati*** | ***tempo totale nel super.*** | ***tempo tot. speso in coda*** | ***n. di code visitate*** |
Per le casse:
| ***id cassa*** | ***n. prodotti elaborati*** | ***n. di clienti*** | ***tempo tot. di apertura*** | ***tempo medio di servizio*** | ***n. di chiusure*** |

I tempi vanno espressi in secondi con al più 3 cifre decimali. Lo studente, se lo ritiene, può arricchire le informazioni prodotte dallo script.

Note finali

Ci si attende che tutto il codice sviluppato sia, per quanto possibile, conforme POSIX. Eventuali eccezioni (come ad esempio l'uso di estensioni GNU) devono essere documentate nella relazione di accompagnamento. Per attendere un certo numero di millesecondi, si può utilizzare la chiamata *nanosleep*. Fare attenzione alla generazione dei numeri casuali: utilizzare differenti seed per ogni thread distinto che invoca la rand_r.
La consegna dovrà avvenire, entro i termini previsti per la consegna in ogni appello, attraverso l'upload sul sito Moodle del corso di un archivio con nome ***nome_cognome-CorsoX*.tar.gz** contenente tutto il codice necessario per compilare ed eseguire il progetto (CorsoX sarà CorsoA o CorsoB a seconda del corso di appartenenza dello studente). Scompattando l'archivio in una directory vuota, dovrà essere possibile eseguire i comandi *make* (con target di default) per costruire tutti gli eseguibili, e *make test1* e *make test2* per eseguire i tests e vederne i risultati. L'archivio dovrà anche contenere una relazione in formato PDF (di massimo 5 pagine – Times New Roman, 11pt, margini di 2cm formato A4, interlinea singola) in cui sono descritti gli aspetti più significativi dell'implementazione del progetto e le scelte fatte. Il progetto può essere realizzato su una qualsiasi distribuzione Linux a 64bit. In ogni caso, i test devono girare senza errori almeno sulla macchina virtuale Xubuntu fornita per il corso e configurata con almeno 2 cores.

Infine, si ricorda che per sostenere l'esame ***è necessario iscriversi*** *sul portale esami.*

PER CHI CONSEGNA IL PROGETTO ENTRO L'APPELLO DI LUGLIO 2020

Per chi consegna il progetto entro la sessione estiva, il progetto che lo studente deve realizzare è **ridotto e semplificato** nel modo seguente rispetto a quanto specificato nelle sezioni precedenti:

1. I clienti in coda in una cassa non si spostano in altre casse (non va implementato l'algoritmo di decisione);
2. Il direttore non è più un processo separato, ma un thread all'interno del processo supermercato. I segnali SIGQUIT e SIGHUP vengono quindi inviati al processo supermercato, e non c'è alcuna interazione via socket AF_UNIX.
3. Il test che ha come target del Makefile *test1* non deve essere realizzato. Il Makefile avrà solamente un target *test* che corrisponde a quanto descritto per il target *test2*.