Relazione Progetto di Laboratorio di Sistemi Operativi

Introduzione

Il programma ha l'obiettivo di simulare il ciclo di lavoro di un supermercato. Si sviluppa su 3 thread principali più 2 di supporto, ad ognuno dei quali viene assegnato un lavoro specifico. All'avvio viene richiesto un file di configurazione da cui vengono estratte le impostazioni con cui dovrà essere eseguita la simulazione. Alla conclusione del lavoro, viene creato un file "s.log" che verrà analizzato dallo script apposito per estrarre i dati e produrre sullo standard output le informazioni relative alle casse e ai clienti.

Supermercato

Il thread principale ha la funzione di avvio, generazione e controllo clienti, infine della chiusura del lavoro.
Si occupa inizialmente della lettura delle impostazioni dal file "config.txt", passato come argomento, e dell'allocazione degli elementi utili all'intero processo. Tra questi si hanno le strutture per le casse, con le relative code, e quelle per i clienti. Per l'acquisizione dei dati di configurazione viene chiamata l'apposita funzione "LeggiConf" che utilizza una "fopen()" dato il file testuale volutamente minimale. Per la creazione delle strutture necessarie all'intero lavoro sono utilizzate varie "calloc" per evitare errori di valori non inizializzati. Viene inoltre inizializzato il seme per la generazione pseudorandomica dei dati dei singoli clienti per evitare intere sequenze di dati uguali, il seme verrà utilizzato in seguito dalla funzione apposita. Viene quindi creato il thread Direttore e inizia la generazione dei clienti. Sfruttando il limite dei clienti possibili all'interno del supermercato e un flag "run", da tre possibili valori per conoscere lo stato del singolo thread cliente, vengono usati due contatori e un puntatore per gestire la condizione degli acquirenti. Il contatore "at" viene utilizzato per la visita lineare dei thread clienti e il "ClIn" per sapere quanti clienti sono attivi. Se si è raggiunto il limite di clienti, il Supermercato rimane in attesa di poterne creare di nuovi. Alla chiusura del supermercato vengono controllati i clienti ed il Direttore, vengono quindi liberate le risorse occupate. In seguito viene creato il log riempito coi dati dei singoli clienti (un id, il tempo trascorso all'interno del supermercato, il tempo trascorso in coda alla cassa e il numero di oggetti acquistati) e delle casse (l'id, il numero di clienti serviti, il tempo totale di apertura, il numero di aperture e il tempo medio di servizio, seguiti dai tempi di servizio e dagli oggetti acquistati dai singoli clienti serviti). Viene infine liberata la memoria e il processo può terminare.

Direttore

L'obiettivo del thread è la gestione le casse. Si avvia con la chiamata alla funzione per il signal handler, essendo un singolo processo multithread, la nuova gestione viene applicata a tutta l'esecuzione e viene eseguita nel momento in cui la ricezione del segnale SIGHUP acquisisce il significato desiderato. Viene poi chiamato un sottoprocesso di supporto "out" per la gestione dell'uscita dei clienti. Nell'interpretazione data, il supermercato apre con una cassa attiva, questa viene subito gestita e si considera sempre aperta, fino alla chiusura del lavoro. Viene utilizzato un vettore "CassaAperta" per individuare, internamente al thread Direttore, le casse attive. Si apre quindi il ciclo infinito basato sulla guardia "dir_OP" che continuerà fino alla ricezione del segnale di terminazione. Se presente il flag "QWm" con la disponibilità di una nuova cassa, questa viene ricercata a partire dalla seconda (la prima è sempre aperta e si fa riferimento ad una situazione dove le casse con id più basso siano le più utilizzate) e viene aperta. In seguito avviene la verifica sulle casse con coda di massimo un cliente, se ne vengono trovate a sufficienza, in accordo col file di configurazione, viene selezionata quella da chiudere a partire dalle ultime (cercando sempre di tenera aperte quelle con id minori). All'uscita del ciclo vengono chiuse tutte le casse, viene liberata la memoria e si attende la chiusura del thread di supporto. Il Direttore può quindi terminare.

Out

Il compito del thread è la chiusura delle operazioni dei clienti e la registrazione dei dati generali collegati ad essi. Esegue un ciclo basato sul flag "exOP" gestito dal Direttore e sulla coda "wait_ex" associata all'uscita dei clienti. La guardia è contraddistinta da un OR per poter permettere, alla chiusura del supermercato, di far terminare i clienti ancora all'interno quando si riceve l'apposito segnale. Gestisce gli oggetti totali venduti con "TotOb" ed il totale dei clienti con "TotCl" ed aggiorna il numero dei clienti all'interno con "ClIn".

Cassiere
Il ruolo del thread è la gestione della singola cassa. Inizialmente viene randomizzato il tempo costante impiegato per ogni cliente e viene avviato il sottoprocesso di supporto “timer_desk”. Viene aperto il ciclo di lavoro e si ha una verifica condizionale utile per la chiusura, senza questa il thread non terminerebbe alla chiusura dettata dal direttore. Il lavoro del Cassiere sfrutta una coda associata con politica FIFO, quando trova una richiesta salva il numero degli oggetti, aggiorna la richiesta e la coda, infine genera i dati per il log. All’uscita dal ciclo vengono aggiornati i dati generali della cassa e si attende la chiusura del thread di supporto. In seguito viene spostata la coda, alla prima cassa (sempre attiva, passandole una coda minima) se il supermercato è ancora aperto o all’uscita in caso di chiusura (in questo caso vengono azzerati gli oggetti del cliente).

timer_desk
Ha il compito di gestire l’aggiornamento della coda segnalata al Direttore. Si basa su un’attesa e una lunghezza della coda, entrambe definite dal file di configurazione, e in caso opportuno segna il flag “QWm” indicando la necessità di apertura di una nuova cassa.

Cliente
Il thread ha il fine di simulare il lavoro di un singolo cliente. Inizia con la randomizzazione dei dati di tempo speso all’interno del supermercato e oggetti da acquistare utilizzando la funzione “randomCL”. Passato il tempo stabilito, se gli oggetti sono maggiori di zero si accoda ad una cassa, questa viene selezionata randomicamente in caso di più casse aperte, altrimenti si accoda direttamente per l’uscita dal supermercato. In entrambi i casi verrà aspettato l’aggiornamento del flag di richiesta “req” dai sottoprocessi a cui rifescono per passare alla fase successiva. Alla fine registra i propri dati per il log e aggiorna lo stato “run” per segnalare la conclusione del lavoro.

Script analisi.sh
Lo script prende come argomento un file .log e ne verifica l’estensione. Avvia il lavoro generando gli array utili per la stampa dei dati e un contatore per le casse. Il file ricevuto viene analizzato sequenzialmente, lo script non si aspetta una sequenza nei clienti, i cui dati vengono subito mandati in output, ma solo sulle casse, queste devono essere seguite dai relativi clienti, anche non in ordine. L’unico calcolo effettuato dallo script riguarda i totali degli oggetti venduti da ogni singola cassa. Al termine dell’analisi vengono mostrati anche i dati delle casse. L’output si conclude con “END TEST”.

Makefile
Il makefile contiene tre target e si basa sulle directory definite. Il target “all” genera il file oggetto, la libreria e l’eseguibile del programma. Il target “clean” ha la funzione di pulizia degli elementi creati da “all” e del log generato dal processo. Il target “test” lancia la simulazione richiesta dal progetto, tra la terminazione di questa, segnalata da “test END”, e l’avvio di “analisi.sh” si ha una “sleep” per evitare errori da parte dello script. Al makefile è inoltre affidata la redirezione dell’output del processo sul file log.

Note finali
I dati di configurazione scelti sono mirati ad evidenziare le varie fasi di lavoro e la differenza che si ottiene con più casse attive, questo va a discapito della chiusura delle casse che comunemente non avviene, dato il lavoro costante del thread Supermercato e la randomizzazione della scelta della cassa da parte dei clienti. La funzionalità è stata comunque testata regolando pesantemente i tempi di attesa, il numero dei clienti attivi, il tempo di aggiornamento della coda comunicata “Cas.QW”, il numero di casse aperte contemporaneamente alla richiesta di una singola, modificando il valore di coda minima e portando ad 1 il numero delle casse necessarie con coda minima per attivare la chiusura.
Per ottenere una equa gestione con un numero di casse aperte maggiore di uno, ancora meglio con più della metà, e facendo riferimento alla scelta di privilegiare le casse con id minore, la randomizzazione sulla scelta della cassa e della relativa coda può risultare molto sbilanciata in caso di test con durata inferiore ai 5 secondi e/o con poche casse operative. Per limitare lo sbilanciamento, nella funzione “randomCas” si ha una ricerca dalla

cassa selezionata, se chiusa, verso un’id minore, sapendo che al caso pessimo la prima sarà sempre aperta. Ci si aspetta comunque un lavoro medio superiore per le casse con id minore.
Il Direttore utilizza il vettore “CassaAperta”, non fa direttamente riferimento ai flag di apertura interni alle casse (Cassa.ap) ma li aggiorna per evitare riferimenti non definiti ed avere accesso esclusivo.
Il numero totale dei clienti del log fa riferimento a quelli attivati e che hanno completato il lavoro, compresi quelli con zero oggetti acquistati e quelli bloccati dalla chiusura del supermercato, ci si aspetta quindi una discrepanza tra la somma dei clienti serviti dalle casse e il numero totale dei clienti registrato nel log.