Relazione progetto SOL

Maurizio Cascino

2019

Indice

1 Introduzione

L'object store è un sistema client-server, realizzato per supportare richieste di memorizzazione, recupero e cancellazione di oggetti.

2 Object store (Server)

2.1 Main Principale

Il *main thread* avrà il compito di controllare eventuali richieste di connessione da parte dei clients. Se la connessione tra server e client avverrà con successo, verrà creato un nuovo thread che gestirà le richieste del client. Tale operazione verrà fatta ogni volta che un client tenterà di connettersi.

La funzione select, impostata con un opportuno timeout, ci permetterà di evitare che il main thread si blocchi sull'accept e di gestire un'eventuale terminazione di quest'ultimo.

Il *main thread* ignorerà e reindirizzerà tutti i *segnali* che riceverà in un nuovo thread (`signal_thread`). Infine quando la variabile globale (*sig_atomic_t*) `quit` verrà settata a 1, vorrà dire che il server avrà ricevuto un segnale di terminazione. Il server, prima di chiudersi, attenderà (con una `pthread_join`) che il `signal_thread` finisca di lavorare. In tal modo i threads che gestiscono i clients avranno modo di terminare l'operazione in corso e quindi, lasciare l'object store in uno stato consistente.

2.2 Signal Thread

Il thread che gestirà eventuali segnali sarà il `signal_thread`. I segnali che gestirà saranno:

- **SIGUSER1.** verranno stampate sullo standard output informazioni riguardanti lo stato del server (numero clients connessi, numero oggetti nello store, size totali dello store). Tali informazioni verranno memorizzate in una struttura dati condivisa dai threads, alla quale potranno accedervi solo in mutua esclusione.
- **SIGTERM, SIGQUIT, SIGINT.** sono segnali di terminazione. Verrà settata a 1 la variabile globale `quit` in modo tale che i threads che gestiscono i clients verranno avvisati che devono terminare il prima possibile, lasciando l'*object store* in un stato consistente. Intanto il `thread_signal` si metterà in attesa, finché non riceverà una signal dall'ultimo thread-client che si sarà disconnesso. A questo punto anch'esso potrà terminare.

2.3 Create_thread

Create_thread si occuperà di gestire le richieste inoltrate dal client. In questo thread, verranno mascherati i segnali. Ogni operazione eseguita dal client avrà un header specifico in modo tale da poter discriminare le operazioni. Le operazioni sono le seguenti: *Register, Store, Retrieve, Delete e Disconnect*. Come da specifica, i nomi degli oggetti memorizzati sono garantiti essere univoci e i nomi degli oggetti sono garantiti essere tutti distinti.

Inoltre se il client non ha eseguito la connessione tramite la Register, le operazioni di Store, Retrieve, Delete e Disconnect non possono essere effettuate.

3 Client

3.1 Client e libreria

Per comunicare con il server, il client avrà a disposizioni le funzioni della libreria `lib.c`. In ogni funzione implementata nella libreria, verranno valutati i parametri affinché questi ultimi siano validi. In caso contrario, verrà settata la variabile `errno` e con un `goto _exit` si terminerà la funzione.

Il client utilizzerà la libreria `lib.c` per effettuare i test richiesti. I test effettuati sono i seguenti:

- **Test 1.** Vengono memorizzati 21 files, di cui 1 è un file di testo, i restanti sono files che rappresentano arrays di interi di dimensione crescente da 100 a 100000 bytes.
- **Test 2.** Vengono recuperati 3 files: `bibbo.txt`, `file_1` e `file_20`.
- **Test 3.** Vengono cancellati i 3 files precedentemente recuperati.

Prima che il client termini, stamperà su standard output l'esito operazioni effettuate.

3.2 Makefile, script e file di log

Nel Makefile ci saranno i target phony: *all*, *clean*, *cleanall* e *test*. Tramite il target test verrà eseguito lo script `testsum.sh` che effettuerà le seguenti operazioni:

1. Verranno eseguiti contemporaneamente 50 istanze del client di test che effettueranno test di tipo 1.
2. Verrà lanciato il segnale `SIGUSR1` al server.
3. Verranno eseguiti contemporaneamente 50 istanze del client di test precedentemente registrati, dei quali 30 effettueranno il test di tipo 2 e 20 il test di tipo 3.
4. Verrà lanciato il segnale `SIGQUIT` al server.
5. Elaborazione `testout.log` e stampa su standard output

3.3 Sistemi Operativi testati

1. Xubuntu
2. Ubuntu