Relazione progetto Laboratorio di Sistemi Operativi a.a. 2018-19

Nicola Venturi

1 Scelte progettuali

In questa sezione sono mostrate le scelte effettuate ed utilizzate durante lo sviluppo di questo progetto.

1.1 Struttura dati di appoggio

Per memorizzare informazioni relative ad un client ho deciso di utilizzare la struttura *worker_t* definita sulla *struct worker*, come mostrato di seguito

```
typedef struct worker{
    struct worker *nxt;
    struct worker *prv;
    int workerfd;
    int connected;
    char _name[MAXNAME+1];
}worker_t;
```

I singoli worker sono collegati tra di loro tramite la lista **worker_l*, che mantiene memorizzati i worker di client che si sono connessi all'ObjectStore.
Il campo *int workerfd* memorizza il file descriptor utilizzato dal client per connettersi alla socket dell'ObjectStore; il campo *int connected* se inizializzato a 0 mostra che il client non è connesso in quel momento all'ObjectStore, mentre se inizializzato a 1 mostra che il client è connesso in quel momento, rendendo così impossibile ad un altro client con lo stesso nome di collegarsi all'ObjectStore; il campo *char _name[MAXNAME+1]* indica il nome con cui il client si è collegato, che può avere dimensione massima di 50b.

1.2 Variabili condivise

All'interno del codice che definisce ObjectStore, thread worker e thread signaller ho utilizzato varibili condivise definite con **worker_l* per definire la lista dei worker, *conn_cl* per definire il numero di client connessi in un determinato momento all'ObjectStore, *tot_size* per definire la dimensione totale dell'ObjectStore, *n_obj* per definire il numero di oggetti presenti nell'ObjectStore, *serveronline* per definire lo stato di esecuzione del server. Tutte le variabili condivise sono state modificate in mutua esclusione.

2 Struttura del codice

Il codice è formato da un server multithread chiamato *Objectstore,* da un client e da una libreria *access* per l'interfacciamento del client con l'Objectstore, attraverso la quale il client può eseguire le operazioni disponibili.

- ***objectstore.c*** implementazione dell'ObjectStore, si occupa di lanciare un thread per la gestione dei segnali, di accettare le connessioni con i client e di lanciare il thread worker relativo al client accettato.

- ***worker.c*** implementazione del thread worker, uno per ogni client, si occupa di ricevere i messaggi dal client ed eseguire le operazioni richieste.

- ***signal_t.c*** implementazione del thread signaller, gestore dei segnali, si occupa della gestione dei segnali inviati all'ObjectStore.

- ***access.c*** implementazione della libreria d'accesso che il client utilizza per interfacciarsi con l'ObjectStore.

- ***client.c*** implementazione del client di test, strutturato secondo le specifiche fornite dal testo del progetto.

- ***utils.h*** header nel quale sono definite *define* e funzioni utili all'interno di ogni altro file descritto precedentemente; questo file contiene le funzioni *writen* e *readn* definite dal professore all'interno del file conn.h della soluzione all'assegnamento 11.

3 Gestione dei segnali

La gestione dei segnali implementata nel file *signal_t.c* è affidata ad un thread *signaller* che gestisce i segnali SIGINT, SIGTERM, SIGUSR1, lasciando gli altri segnali ad una gestione di default.

Alla ricezione dei segnali SIGINT e SIGTERM il thread signaller stampa sullo stdout la ricezione del segnale e setta a 0 la variabile condivisa *serveronline* così da fare terminare ogni thread woker attivo e l'ObjectStore stesso.

Alla ricezione del segnale SIGUSR1 il thread signaller invocando la funzione *s_print_report* stampa un report sullo stato dell'ObjectStore, stampando il contenuto delle variabili condivise *conn_cl, n_obj* e *tot_size,* quest'ultima stampando la dimensione in Kb o Mb.

4 Struttura libreria di accesso

La libreria di accesso mette a disposizione del client i seguenti metodi

- ***int os_connect(char *name)*** – inizia la connessione all'ObjectStore, registrando il client con il nome *name.* Se la connessione ha avuto successo ritorna *true,* altrimenti *false.*

- ***int os_store(char *name, void *block, size_t len) –*** richiede all'ObjectStore la memorizzazione dell'oggetto puntato da *block,* per una lunghezza *len*, con il nome *name.* Restituisce *true* se la memorizzazione ha avuto successo, *false* altriementi.

- ***void *os_retrieve(char *name) –*** recupera dall'ObjectStore l'oggetto precedentemente memorizzato sotto il nome *name*. Se il recupero ha avuto successo, restituisce un puntatore a un blocco di memoria, allocato dalla funzione, contenente i dati precedentemente memorizzati. In caso di errore, restituisce NULL.

- ***int os_delete(char *name)*** – cancella l'oggetto di nome *name* precedentemente memorizzato. Restituisce *true* se la cancellazione ha avuto successo, false altrimenti.

- ***int os_disconnect()*** – chiude la connessione all'ObjectStore. Restituisce *true* se la disconnessione ha avuto successo, *false* in caso contrario.

5 Esecuzione e test

5.1 Esecuzione

Eseguire il comando *make* per generare gli eseguibili. Per eseguire l'ObjectStore lanciare in una shell il comando *./objectstore.o*, mentre in un'altra shell lanciare il comando *make test.*
Per lanciare il segnale SIGUSR1, come riferito dal professore Prencipe in aula duramente la lezione di spiegazione del testo, diversamente da quanto invece scritto sul testo del progetto, è stato implementato il comando *make sigusr1.*
Per far terminare l'ObjectStore con il segnale SIGINT è stato implementato il comando *make exit1,* mentre per farlo terminare con il segnale SIGTERM il comando *make exit2.*
Per lanciare lo script di analisi testsum.sh che calcola e stampa su stdout il contenuto del file testout.log è stato implementato il comando *make test.*
Infine per cancellare gli eseguibili, la cartella *data* contenente i dati dei client, il file testout.log e la libreria di accesso libaccess.a è stato implementato il comando *make clean.*

5.2 Test

Il progetto è stato testato sui seguenti sistemi operativi:

- Ubuntu 18.04 LTS
- Debian 9.9
- Xubuntu 14.4 (VM fornita dal docente)