Sistemi Operativi 1

A. A 2012/2013

Progetto 2 – Threads

Alberto Buratti, Lorenzo Lotto Mirko Morandi

3 Giugno 2013

Approccio architetturale:

la soluzione consiste di un eseguibile multithreaded che implementa funzionalità di logging e strutture dati thread-safe

Eseguibile:

- Deve ricevere in input una stringa di testo da standard input, cifrarlo, decifrarlo e riproporlo nello standard output; l'applicazione deve terminare con l'inserimento di una determinata sequenza di caratteri
- Ogni operazione che viene eseguita è affidata ad un thread dedicato, la cui esecuzione viene sbloccata da un'apposita segnalazione da parte del thread precedente.
- Lo scambio di dati tra thread avviene tramite una serie di code FIFO thread-safe, che implementano quindi controlli per regolarne la concorrenza.
- Ad ogni inserimento/estrazione in una delle code viene controllato il buon esito dell'operazione. Se fallisce l'errore viene segnato nel file di log e l'applicazione termina con apposito status code

Logger:

- Le funzioni di creazione e di scrittura dei file di log vengono implementate utilizzando il servizio di logging offerto dal sistema operativo, nella fattispecie syslog
- Vi sono quattro file di log, uno per ogni thread (tr.log, te.log, td.log e tw.log), che vengono creati nella cartella /var/log/threads/ al momento dell'installazione o in seguito alla loro eliminazione
- Sono state create tre funzioni, a seconda del tipo di informazione che viene registrata nel log, alle quali vengono passati come parametri la stringa da scrivere nel file di log e il nome del thread nel quale si è verificato l'evento
- Un timestamp del momento nel quale un log è stato scritto viene automaticamente posizionato davanti alla stringa di log
- Per evitare accessi concorrenti allo stesso log da parte dello stesso processo, la scrittura su log di un evento viene effettuata bloccatando tramite mutex l'accesso alla funzione di log per tutto il tempo necessario a collegarsi al servizio di log, scrivere il record e scollegarsi dal servizio

Queue:

- Implementazione di una coda di tipo FIFO thread-safe, tramite l'utilizzo di un mutex sulle operazioni di aggiunta e rimozione di elementi dlla coda.
- La coda è implementata come una linked list di struct costituiti da: un array di char che contiene i dati necessari al thread, un puntatore a struct che referenzia la componente precedente della lista e un puntatore a struct che referenzia la componente successiva nella lista.

Makefile:

- È stato creato un makefile che si occupa di automatizzare la compilazione dei file sorgente e la confiugurazione del servizio di log. (sono richiesti i privilegi di root per configurare il servizio di log)
- La compilazione viene effettuata con le opzioni -Wall (in modo da mostrare tutti i warning)

- È stata prevista la possibilità di effettuare un'installazione dell'applicazione system-wide (sono richiesti privilegi di root durante questa fase). Il comando make install provvederà a copiare i file eseguibili nelle directory corrette e imposterà i permessi di esecuzione per gli utenti
- Il makefile fornisce inoltre la funzione 'clean', che permette di ripulire la directory contenente i file sorgente dai file oggetto e dagli eseguibili creati durante la compilazione.

Scenario:

Lo scenario di utilizzo tipico prevede l'applicazione in esecuzione su un terminale in foreground. Durante l'installazione tramite makefile vengono create le regole necessarie a syslog per la corretta suddivisione dei file di log. L'applicazione, una volta avviata, inizializza le proprie code e si pone in attesa di input da parte dell'utente. Quando l'utente fornisce una stringa all'applicazione, questa provvede a crittografarla e a mostrare la stringa crittografata, viene poi decrittata e il risultato finale viene mostrato in output.

Segue un'analisi dettagliata del funzionamento:

- Inizializzazione delle strutture dati e dei semafori
- Il thread che si occupa di leggere input dall'utente riceve una sequenza di caratteri, questa viene controllata per assicurarsi che non sia la stringa di uscita e viene copiata nella coda di input. Viene infine effettuata una signal al semaforo usato dal thread successivo per iniziare a processare i dati, e il thread si rimette in attesa di input da parte dell'utente
- Il thread che si occupa di cifrare l'input riceve la signal dal thread precedente, che ne sblocca l'esecuzione. Prosegue estraendo la stringa letta in precedenza e andrà a leggere una sequenza di byte da /dev/random di pari lunghezza. Tale sequenza sarà poi convertita in una stringa formata da caratteri leggibili, ovvero dai caratteri 33 a 126 secondo la tabella ASCII. La stringa ottenuta verrà poi usata per generare una stringa cifrata tramite xor byte per byte. Dopodichè il thread inserirà entrambe le stringhe nelle rispettive code edeffettuerà una signal al thread successivo e riprenderà il processo dall'inizio
- Il thread che si occupa di decifrare i dati estrae la stringa cifrata dalla rispettiva coda e ne applica una xor, ottenendo la sequenza di caratteri iniziale. Il risultato viene inserito nella rispettiva coda e viene segnalato al thread finale che sono presenti dati da stampare a video. Dopodichè processa i dati rimanenti, se non ve ne sono si mette in attesa
- Il thread di stampa estrae la sequenza di caratteri dalla coda dei risultati e la stampa sullo standard output, rimettendosi subito in attesa di altre sequenze da stampare, se non ve ne sono
- Quando il thread iniziale riceve in input la sequenza di terminazione "quit", i thread vengono sbloccati e, trovando le strutture dati da cui dipendono vuote, terminano la loro esecuzione.

Comandi:

Per compilare il progetto, l'elenco di comandi da eseguire in una shell è il seguente:

- cd /directory_del_progetto/ (a questo punto, l'esecuzione del comando 'ls' dovrà fornire un elenco contenente una directory "src" e un Makefile)
- make (come root se si desiderano i log separati)
- make install (opzionale, come root, effettua l'installazione system-wide)

A questo punto l'applicazione è stata installata a livello di sistema, ed è quindi disponibile per tutti gli utenti. Nel caso si sia scelto di non effettuare l'installazione system-wide, sarà necessario spostarsi nella directory denominata "src" ed eseguire localmente l'applicazione, digitando

- ./threads

Nel caso invece di un'installazione system-wide, l'eseguibile sarà copiato nella directory /usr/bin/ e sarà utilizzabile semplicemente digitando il comando "threads" in una shell

Autori:

Il progetto è stato sviluppato da Alberto Buratti (145552), Lorenzo Lotto (151775) e Mirko Morandi (151778)

Contatti:

- Alberto Buratti alberthohenstaufen@gmail.com
- Lorenzo Lotto llorenzo@outlook.com
- Mirko Morandi nanotrippolo@gmail.com