Relazione Progetto di Sistemi Operativi

Servizio di calcolo dell'impronta SHA-256 mediante Pthread e FIFO

Alessia Gennari - VR488137

Agosto 2025

Lobiettivo del progetto è la realizzazione di un server in grado di eseguire più calcoli di impronte SHA-256 in parallelo, e di un client che invii al server le informazioni sui file di input, ricevendo in risposta l'impronta calcolata. Per l'implementazione sono stati utilizzati `Pthread` e `FIFO`.

1 Struttura del progetto

Il progetto è organizzato in tre cartelle principali:

- **src**: contiene tutti i file sorgente
  - **server.c**: gestisce la ricezione delle richieste, l'inserimento in coda e l'invio delle risposte.
  - **client.c**: invia al server le richieste di calcolo dell'impronta SHA-256 e riceve la risposta.
  - **cache.c**: implementa le funzioni per l'inserimento, l'eliminazione e la gestione delle notifiche nella cache.
  - **queue.c**: implementa le funzioni per l'inizializzazione, l'inserimento, la rimozione e l'eliminazione di elementi nella coda.
- **include**: contiene i file di intestazione
  - **common.h**: definisce le strutture `request` e `response`, oltre a dati costanti.
  - **cache.h**: definisce la struttura dell'elemento di cache e dichiara le funzioni di gestione.
  - **queue.h**: definisce la struttura della coda e dichiara le funzioni di gestione.
  - **file_to_hash.txt**: file di test.
- **tmp**: contiene principalmente il file *fifo_request.txt*.

2 Funzionalità implementate

- Server che riceve richieste e invia risposte tramite `FIFO`.
- Client che invia richieste e riceve risposte tramite `FIFO`.
- Gestione di richieste simultanee con schedulazione basata sulla dimensione del file.
- Thread pool fisso di 3 thread in attesa sulla coda di richieste.
- Meccanismo di caching per evitare ricalcoli in caso di richieste ripetute.
- Chiusura di tutti i worker e del file `fifo_request` alla chiusura del server.

3 Scelte progettuali e difficoltà incontrate

L'implementazione del codice è stata realizzata per fasi, in modo da garantire una base solida ad ogni passaggio. In un primo momento è stata progettata la struttura di base di server e client, successivamente sono stati introdotti i thread e, infine, la cache con il relativo meccanismo di interrogazione.

I file `FIFO` sono stati creati nella cartella `/tmp` e vengono eliminati al termine dell'esecuzione.

Una delle difficoltà riscontrate è stata la gestione della lettura delle richieste nel server. In una prima implementazione, si era utilizzato un buffer con troncamento per separare `pid` e nome del file, ma ciò causava ritardi ed errori. La soluzione adottata è stata l'introduzione di due strutture dedicate per richieste e risposte, con il campo `pid` di tipo `pid_t` anziché `char`. Questa scelta ha migliorato le prestazioni e ridotto gli errori dovuti a variazioni nella lunghezza della stringa.

Un'altra difficoltà è stata la gestione degli accessi alla cache quando più thread cercano di accedervi. In una prima stesura approssimativa, a causa di alcune dimenticanze, infatti è capitato che desse errore.

4 Compilazione ed esecuzione

La compilazione avviene tramite `CMake`:

1. Creare la cartella `build`.
2. All'interno di `build`, eseguire:

   ```
   cmake ..
   make
   ```

3. Avviare il server:

```
./server
```

4. In un altro terminale, avviare il client:

   ```
   ./client ../include/file_to_hash.txt
   ```