RELAZIONE HOMEWORK 2

Antonio Pietro Romito mat.1932500

Sistemi Operativi II A.A. 2022

COMPILAZIONE ED ESECUZIONE DEI PROGRAMMI

Per **compilare** i programmi digitare il comando: > *make*
O in alternativa i comandi: > gcc -pthread server.c -o server
> gcc client.c -o client

Per **eseguire** il programma **server** digitare il comando (in background o su una shell diversa dal client): > *./server*
Per il **client** invece (dopo aver avviato il server): > *./client*

SERVER.C

Come prima cosa il programma client crea un **socket** (si veda la funzione *create_socket*) in ascolto sulla porta 60000, inizializza il semaforo utilizzato per accedere concorrentemente al file di log e viene aperto per l'appunto il file.

Successivamente entra in un **ciclo infinito** (il server deve essere sempre attivo per accettare nuove connessioni). All'interno del ciclo vengono **accettati i client** che chiedono di connettersi al server, creando un nuovi socket per le singole connessioni. Per ogni client viene **creato un nuovo thread** che gestisce la connessione a cui viene passato come argomento il puntatore ad una struttura contenente il socket descriptor della connessione, il file descriptor del file di log e la stringa che rappresenta l'indirizzo ip del client.

Ogni thread gestisce l'interazione con il client, quindi quando viene creato il thread parte l'esecuzione parallela dalla funzione *start*. In essa è definito un ciclo che può essere terminato solo quando viene letta la lettera 'q' dal socket. Una volta usciti dal ciclo viene **chiuso il socket** e **terminata l'esecuzione del thread**.

Ad ogni iterazione del ciclo viene **letto il messaggio inviato dal client** dal socket e **calcolato il timestamp** corrente (ossia quello della ricezione della richiesta).

Successivamente viene **calcolato il risultato dell'operazione**, attraverso la funzione *calc*, la quale scrive su un buffer la stringa contenente il risultato, che in caso di successo rappresenta il numero reale, risultato dell'operazione, altrimenti un carattere che codifica l'errore. Gli errori possono essere: di formattazione (carattere f), se la richiesta del client non è formattata nel seguente modo: *[operazione, operando1, operando 2]*, di divisione per 0 (carattere d), se viene richiesta una divisione con divisore uguale a 0 e di operatore non valido (carattere o), se l'operazione richiesta non è tra quelle possibili (+ - * /).

Dopo viene **calcolato un nuovo timestamp** (di invio della risposta) e viene **scritto il messaggio di risposta sul socket** nel formato *[timestamp ricezione richiesta, timestamp invio risposta, risultato operazione]*. Dei timestamp vengono presi in considerazione solo i nanosecondi dato che il tempo di servizio, ossia la differenza tra il tempo di invio della risposta e quello di ricezione della richiesta, sarà sicuramente inferiore ad un secondo. Infatti la struttura utilizzata (*timespec*) comprende due valori per rappresentare il tempo: il tempo in secondi trascorso dalla mezzanotte del primo gennaio 1970 e i restanti nanosecondi.

Infine, l'ultima operazione presente nel ciclo è la **scrittura dell'operazione effettuata su file di log** attraverso la funzione *log_write*: essa blocca il semaforo prima di scrivere sul file e lo sblocca una volta avvenuta la scrittura ed effettuato il flush, in modo da garantire l'accesso a tutti i thread concorrenti. Sul file di log viene scritta **una riga per ogni operazione** con i seguenti dati: l'indirizzo IP del client che ha effettuato la richiesta, l'input inserito dall'utente, il risultato o l'errore generato dall'elaborazione dell'operazione e la data e l'ora in cui è arrivata la richiesta al server.

CLIENT.C

La prima cosa che il programma client effettua è la creazione del **socket** (si veda la funzione *create_socket*) ed esso viene **connesso al localhost** sulla porta definita dal server (60000). Successivamente entra in un **ciclo** che può essere terminato solo dalla digitazione della lettera 'q' sullo standard input da parte dell'utente.

Come prima operazione del ciclo viene **letta la stringa inserita dall'utente** attraverso la funzione *input*, la quale, dopo aver letto dallo standard input, **prepara la stringa nel formato corretto** per l'invio al server.

Successivamente **viene inviato il messaggio al server** nel formato *[operazione,*

operando1, operando 2], scrivendolo sul socket. Una volta arrivato viene **letto dal socket il messaggio di risposta** del server nel formato *[timestamp ricezione richiesta, timestamp invio risposta, risultato operazione]*.

Infine viene **stampato l'output** (si veda la funzione *output*) in un formato user friendly, contente il **risultato dell'operazione**, o una stringa di errore in caso non sia stato possibile effettuare il calcolo, e il **tempo di servizio** in nanosecondi.

Come già detto il ciclo termina quando l'utente digita 'q' e a quel punto viene **chiuso il socket descriptor e terminato il programma**.

TEST

I risultati di alcuni test, delle tipologie indicate nei primi tre punti del piano test della consegna, sono presenti nel file 'log.txt'.

Il quarto ed il quinto punto, ossia se il client non riesce a connettersi al server perché non attivo e la terminazione inaspettata del server, vengono soddisfatti stampando sullo standard error un errore e terminando il programma.