Relazione

Cognome: Perera | Nome: Rideewitage Lachitha Sangeeth | Matricola: 2042904

Introduzione

Per la crezione dell'homework ho realizzato:

1. un file `header.h` in cui ho specificato due struct (utilizzate per il salvataggio delle informazioni sia dei messaggi che riceve il server che quelli che invia il client)
2. un file `server.c` che permette ai client di comunicare con il server attraverso il comando `./server -a address -p port -d root_dir`
3. un file `client.c` che permette al client di inviare i messaggi al server ed eseguire il trasferimento dei file in una delle due modalita': `-r` o `-w` ; o di ottenere il contenuto di una directory del server attraverso il comando `./client -l -a address -p port -f remote_path`
4. un file `funciton.c` in cui sono implementate delle funzioni utilizzate sia nel client che nel server per il controlle e la creazione dei percorsi e file con cui eseguire il trasferimento

HEADER.H

Per quanto riguarda le struct utilizzate, sono definite nell'header e sono utilizzate per tenere traccia delle informazioni ricevute dal server e inviate dal client:

1. `struct client_inf` : struct in cui sono salvate le informazioni che il client invia al server, salvate nel modo seguente:
    - `char* ip_server`
    - `int portno_server`
    - `char* loacl_path`
    - `char* loacl_file_name`
    - `char* remote_path`
    - `char* remote_file_name`
    - `bool read`
    - `bool write`
    - `bool ls_la`
2. `struct msg` : struct in cui sono salvate le informaizoni dei messaggi che il server riceve dal client salvate nel modo seguente:
    - `char* path`
    - `char* file_name`
    - `int mode`

Nell'header sono anche definite funzioni utilizzate sia nel file `server.c` e `client.c` , ovvero:

- `int directory_exist(char* file_name, char* path)` : funzione che dati in input un nome di un file e un percorso controlla se esiste il percorso specificato e se il file specificato esiste all'interno della directory
- `int count_file(char* path, char* file_name)` : funzione che dati in input un percorso e un file conta quanti file con il nome del file passato in input esistono nel percorso passato in input
- `char* create_file(char* file_name, char* path)` : funzione che dati in input un file e un percorso crea il file passato in input nel percorso passato in input
- `void error(char *msg)` : funzione che prende un messaggio di errore e ne lo stampa a schermo con il relativo errore, fermano il programma in cui viene chiamata la funzione

Tali funzioni sono implementate nel file `function.c`

SERVER.C

Il file viene utilizzato per creare la connessione lato server utilizzando i socket come visto a lezione; tuttavia, quando viene usato il comando lato server viene specificato l'indirizzo IP e la porta in cui viene aperto il server e una dir in cui vengono prendere o caricare i file specificati dal client. Una volta che viene impostata la connessione il server rimane in attesa di una richiesta client, una volta che arriva la richiesta da parte di un client, viene istanziato un thread con la socket che si e' collegato al client in modo da gestire piu' client concorrentemente. Il thread, una volta istanziato, chiama la funzione `handle_client(void* socket_desc)` il quale gestisce le richieste del client.
La funzione `handle_client` per gestire le richieste utilizza una variabile come contatore per capire quale messaggio ha ricevuto dal client, esempio: se il contatore sta a 0 allora il server e' in attesa da parte del client la modalita' specificata dal client (se trasferire o ricevere il file oppure eseguire il comando ls -la).
Per fare in modo che il server riceva correttamente il messaggio del client si utilizza un buffer nel quale venogno salvare le informazioni sotto forma di cstringhe.
Un'altra funzione importante implementata all'interno del file e' `mod_write(struct msg cli_msg)` che viene chiamata solo nel caso in cui il client ha specificato la modalita' '-w'; la funzione prende il messaggio del client e ne ricava il percorso il nome del file in cui salvare il file inviato dal client. Se il percorso non esiste o non esiste il file vengono create utilizzando le funzioni definite nell'header e implementate nel file `function.c` . Se il file esiste allora crea il file con un contatore che indica il numero del file (esempio: nel percorso specficato esiste il file `file_nome` , se il client vuole salvare il file sul server con nome del file `file_nome` allora viene creato il file `file_nome(1)`).

CLIENT.C

Per quanto riguarda il file utilizzato per il lato client, il main esegue la connessione al server con l'indirizzo IP e la porta specificato nel comando compilato. Oltre all'indirizzo e alla porta del server a cui collegarsi, il client prende come informazioni la modalita' con cui contattare il server, il percorso locale e anche il percorso remoto (facoltativo).
La funzione inizialmente prende le informazioni da `argv` e le salva nella struct client_inf; in seguito, esegue la connessione con il server prendendo la porta e l'indirizzo che sono salvati nella struct. Una volta eseguita la connessione, invia i vari messaggi al server con la modalita' con cui lavorare (read, write o ls -la); il

remote_path e il remote_file. Per fare in modo che il server capisca quale informaizoni deve andare a prendere ad ogni messaggio, viene usato un contatore anche lato client per fare in modo che quando il server invia un messaggio di risposta sa come gestire la risposta e nel caso continuare con la comunicazione.
Descrizione delle modalita':

1. **Read**: nel caso di questa modalita', il client contatta il server con il file che vuole salvare sul proprio sistema. In questo caso quando il client chiama il comando per comunicare con il server viene effettuato un controllo su `local_path` e `local_file_name` per verificare l'esistenza del percorso e del file all'interno del percorso specificato. Se non esistono vengono creati; sia il percorso che il file. Nel caso in cui il file esiste allora viene creato il file con il nome specificato con un contatore che indica il numero dei file con quel nome. Al server viene solo inviato il `remote_path` e il `remote_file_name` che indicano il percorso e il nome del file che il client vuole salvare sul proprio sistema.
2. **Write**: nel caso della modalita' write, il `remote_path` e il `remote_file_name` vengono creati se non esistenti con gli opportuni controlli eseguiti anche per la modalita' read dal client con il `local_path` e il `local_file_name` .
3. **ls -la**: in questa modalita' viene solo specificato un `remote_path` da cui si vogliono ricavare le informazioni con il comando `ls -la` eseguito con `popen` il quale esegue il comando e salva il risultato del comando su una variabile `FILE*` dalla quale vengono prelevate le informaizoni e mandate al client riga per riga. Una volta che arrivano al client vengono stampate a schermo.

Per l'invio e la ricezione del contenuto del file viene utilizzato un **buffer** in cui vengono salvati i messaggi e inviati con la funzione `send()` il quale invia il messaggio al server; nel caso specifico dei file nel buffer viene memorizzato il contenuto del file riga per riga. Per quanto riguarda la ricezione del file, una volta che il file viene creato nel percorso specificato viene aperto con la funzione `fopen()` il quale aprte il file e lo salva in una variabile `FILE*` ; una volta salvato il file come variabile viene utilizzata la funzione `fpritnf()` .
Una vota che il client o il server finisce di inviare il contenuto del file esegue la funzione `close()` per chiudere il file in modo che viene salvato correttamente.

FUNCTION.C

Per quanto riguarda questo file, sono definite funzioni usate sia lato client che lato server per gestire il controllo/la creazione dei file.

1. **Controllo**: per quanto riguarda il controllo dei file viene usata la funzione `directory_exist` il quale prende un percorso ed eseguendo la funzione `stat()` controlla se riesce a prendere correttamente le informazioni relative al percorso; nel caso in cui non sono state prese correttamente allora il percorso non esiste.
   Nel caso dei file viene effettuato lo stesso controllo passando come parametro in input il percorso con aggiunta del nome del file.
2. **Creazione**: nel caso delle directory, si prende il percorso specificato dal client (uguale sia per `local_path` che `remote_path`); esegue `strtok(path, "/)` in modo da prendere una diredctory alla volta controllando pezzo per pezzo fino a che punto esiste il percorso, nel caso ad un certo punto la parte della directory appena controllata non esiste, allora viene creata con la funzione `mkdir(name_dir, 0755)` .
   Dopo la creazioni delle directory, viene creato il file prendendo i percorso specificato dal client, controlla quanti file esistono con lo stesso nome e crea il file nel percorso specificato con la funzione `fopen(file_name, "w")` e `close(file_name)` per salvare il file nel percorso specificato.