Relazione

I programmi client e server implementano un sistema di trasferimento file che consente operazioni di upload, download e visualizzazione della lista dei file. Il server è progettato per gestire più connessioni client simultanee utilizzando thread. Entrambi i programmi utilizzano socket TCP per la comunicazione di rete e includono funzioni per la gestione dei percorsi dei file, la sincronizzazione e la gestione degli errori.

Programma Server

Struttura del programma

Il programma server utilizza socket TCP per la comunicazione di rete. Il flusso principale del server si divide nelle seguenti fasi:

1. **Analisi degli Argomenti passati in input**: Il server analizza gli argomenti passati in input con getopt per ottenere l'indirizzo IP, la porta su cui ascoltare e la directory root dei file. Se mancano argomenti essenziali, il programma termina con un messaggio di errore.
2. **Creazione e Configurazione del Socket**: Il server crea un socket TCP con socket(AF_INET, SOCK_STREAM, 0). Imposta le opzioni del socket, ad esempio la possibilità di riutilizzare l'indirizzo e la porta. Configura la struttura sockaddr_in con l'indirizzo IP e la porta forniti e lega il socket a questo indirizzo con bind.
3. **Ascolto delle Connessioni**: Il server chiama listen per mettere il socket in modalità di ascolto, consentendo fino a 4 connessioni simultanee.
4. **Gestione delle Connessioni Client**:Il server entra in un ciclo infinito dove accetta le connessioni in arrivo con accept. Per ogni connessione, crea un nuovo thread utilizzando pthread_create, passando la funzione thread_function come funzione di gestione del client. Inoltre, la risorsa associata al thread viene rilasciata automaticamente dal sistema operativo non appena il thread termina la sua esecuzione.
5. **Gestione delle Richieste**: I thread creati per ogni client gestiscono le richieste in arrivo, i comandi vengono interpretati e vengono chiamate le funzioni appropriate attraverso una comparazione (function_for_upload, function_for_download, function_to_send_file).
6. **Gestione delle Richieste del Client**: La funzione thread_function legge i comandi dal client, esegue le operazioni richieste (upload, download, lista file) e invia le risposte al client. Le funzioni di supporto includono:
    - function_for_upload: Gestisce l'upload di file dal client al server, verificando il percorso e scrivendo i dati ricevuti nel file.
    - function_for_download: Gestisce il download di file dal server al client, leggendo i dati dal file e inviandoli al client.
    - function_to_send_file: Invia al client l'elenco dei file presenti nella directory specificata.
    - create_dir: Crea una directory e tutte le sue sottodirectory, se non esistono già.
    - rimuovi_accapo: Rimuove i caratteri di nuova linea da una stringa.
    - Concorrenza: Le funzioni get_file_lock, release_file_lock, lockfile, unlockfile gestiscono un meccanismo di lock dei file utilizzando mutex per garantire la mutua

esclusione durante l'accesso ai file (viene utilizzata una lista puntata file_locks per mantenere traccia dei lock associati ai file)

- get_file_lock cerca un lock associato al file, se non lo trova ne crea uno nuovo
- release_file_lock per il rilascio del lock su un file
- lockfile acquisisce un lock su di un file
- unlockfile rilascia un lock su di un file

Programma Client

Struttura del programma

Il programma client è scritto in C e consente di inviare richieste di upload, download e lista file al server. Il flusso principale del client si divide nelle seguenti fasi:

1. **Analisi degli Argomenti della Linea di Comando**: Il client utilizza getopt per analizzare gli argomenti della linea di comando e memorizza l'indirizzo IP del server, la porta, il comando da eseguire e i percorsi dei file locali e remoti. Se mancano argomenti essenziali o vengono inseriti più comandi contemporaneamente, il programma termina con un messaggio di errore.
2. **Creazione della Connessione**: Il client crea un socket TCP con socket(AF_INET, SOCK_STREAM, 0). Utilizza gethostbyname per ottenere l'indirizzo IP del server a partire dal nome del server. Configura la struttura sockaddr_in con l'indirizzo IP e la porta del server, e si connette al server con connect.
3. **Esecuzione dei Comandi**: Il client invia il comando specificato al server utilizzando la funzione send_command. Le funzioni di supporto includono:
   - upload_file_on_server: Carica un file locale sul server, inviando i dati del file.
   - download_file_from_server: Scarica un file dal server e lo salva localmente.
   - file_list_on_stout: Riceve e stampa l'elenco dei file presenti sul server.
   - create_dir: Crea una directory locale e tutte le sue sottodirectory, se non esistono già.
   - send_command: Invia un comando specifico al server con il path tramite il socket.

Funzioni Condivise

Il programma contiene anche un file con le funzioni condivise da entrambi i programmi come:

1. check_dir: verifica se il percorso inserito è una directory
2. Extract_directory_path: se il percorso non è una directory possiamo procedere con estrarre il percorso per poter eventualmente creare le sottodirectory necessarie(se non esistono)
3. is_valid_port, is_valid_ip_address: gestisce il caso in cui l'indirizzo IP o la porta non siano validi
4. open_file: apre il file specificato con la modalità specificata dall'utente. Verifica se l'apertura del file avviene con successo