README

UDP-ReliableFileTransferProtocol per Ingegneria di Internet e Web 2022-2023

di Lorenzo Casavecchia **<** lnzcsv@gmail.com **>**

Presentazione del progetto

Questo progetto consiste nella realizzazione di un sistema per il trasferimento affidabile di file tra un client e un server, utilizzando come protocollo di trasporto UDP e rispettando le linee guida imposte per il progetto previsto nel corso di **Ingegneria di Internet e Web dell'A.A. 2022-2023**

In questo archivio sono presenti:

- la descrizione della soluzione proposta e della sua implementazione (vedere `DESCRIPTION.md` sotto la cartella `doc`)
- i codici sorgenti utili per l'esecuzione di entrambi client e server (sotto la cartella `src`)
- una descrizione dei problemi architetturali e implementativi riscontrati fino la versione corrente del sistema (vedere `ISSUES.md` sotto la cartella `doc`)
- la traccia originale della consegna del progetto (vedere `ASSIGNMENT.md` sotto la cartella `doc`)

Servizi offerti

Come da specifica, l'applicazione permette il trasferimento file tra un client e un server
Le richieste, generate sempre dal client, possono essere di tipo

- `list` per richiedere una lista dei file correntemente posseduti dal server
- `get <nome>` per richiedere il file `<nome>` correntemente posseduto dal server
- `put <nome>` per caricare il file `<nome>` posseduto dal client, nella cartella dei file del server

Installazione

Per la generazione degli eseguibili è necessario:

1. Cambiare cartella di lavoro ad `src`, per esempio eseguendo

```
cd ./src
```

2. Eseguire

   - `make client` o `make clientd` per la compilazione dell'applicazione client
   - `make server` o `make serverd` per la compilazione dell'applicazione server

   dove `client` e `server` corrispondono a versioni del client e server che non gestiscono dinamicamente i timeout di ricezione / spedizione, mentre `clientd` e `serverd` prevedono l'aggiornamento dei timeout in base all'evoluzione dei ritardi misurati da client e server

   È possibile inoltre generare tutti gli eseguibili con `make all`, che risulterà nella presenza dei file `client`, `clientd`, `server` e `serverd` all'interno della cartella `src`

   Infine tutti gli eseguibili possono essere rimossi con `make clean`
3. Avviare gli eseguibili così generati (`./client` o `./clientd` per il client, `./server` o `./serverd` per il server)
   All'interno di una stesso spazio di lavoro è possibile generare ed eseguire ambi client e server in quanto il codice ed i file previsti per il funzionamento del client e del server risiedono rispettivamente nelle cartelle `./src/client-side` e `./src/server-side`
4. Nel caso sia stata avviata una versione del client (`./client` o `./clientd`) sarà allora possibile inviare una richiesta al server eseguendo

   - `list` per ottenere una lista dei file nella cartella del server
   - `get <nome>` per ottenere dalla cartella del server il file `<nome>`
   - `put <nome>` per caricare nella cartella del server il file `<nome>`

   Nello specifico `list` genererà un file `list.txt` nella cartella `./src/client-side/server_files`

   I file (o la lista dei file) richiesti dal client verranno salvati in `./src/client-side/server_files` mentre i file caricati dal client tramite `put` dovranno essere salvati in `./src/client-side/server_files` prima dell'esecuzione del comando

   Questo significa che per il client tutti i file, caricati o scaricati, risiederanno nella cartella `./src/client-side/server_files`

   Un simile ragionamento è applicato al server: qualora venisse eseguita una versione del server

   - tutti gli aggiornamenti dei suoi file dovuti da richieste `put` di un client saranno visibili nella cartella `./src/server-side/server_files`

- tutte le richieste di tipo `get` effettuate da un client dovranno essere precedute dal caricamento in `./src/server-side/server_files` dei file custoditi dal server
- nel caso di richieste `list` in `./src/server-side/server_files` verrà creato un file `list.txt` contenente la lista dei file correntemente posseduti dal server

  Nel caso in cui venga effettuata una richiesta di `get` di un file non posseduto dal server, il server notificherà il client con un codice di errore (per ulteriori dettagli consultare `DESCRIPTION.md`)

5. Per terminare l'applicazione premere `<Ctrl>+C`

Altri parametri di esecuzione come la durata del timeout, la politica di aggiornamento del timeout e molti altri possono essere manualmente modificati dal file `defs.h` nella cartella `./src/client-side` o `./src/server-side` rispettivamente per le applicazioni client e server

I file `defs.h` sono correntemente identici ma logicamente distinti: applicando modifiche alla `defs.h` del client non porterà cambiamenti alla `defs.h` del server (e viceversa)

Sistema

L'applicazione è stata sviluppata e provata con le seguenti specifiche di sistema

- sistema operativo `Xubuntu 22.04.2 LTS x86_6`
- processore `AMD A8-7410 APU with AMD Radeon R5 Graphics (4) @ 2.200GHz`
- compilatore `gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0`
- debugger `GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1`
- memory error detector `valgrind-3.18.1`
- content tracker `git version 2.34.1`
- grafici `gnuplot 5.4 patchlevel 2`

Servizi non previsti

1. Il sistema non prevede la visualizzazione a schermo dei file scambiati: terminata l'applicazione dovrà essere l'utente ad aprire e visionare il contenuto dei file in questione, per esempio eseguendo

   ```
   less ./src/client-side/server_files/<nome>
   ```

   oppure

   ```
   less ./src/server-side/server_files/<nome>
   ```

   rispettivamente per client e server

2. Il sistema inoltre non preve la creazione o il caricamento di file da tastiera a tempo di esecuzione: il caricamento dei file in `server_files` deve essere effettuata prima dell'avvio dell'applicazione per esempio copiando un file da un altra cartella
3. Il sistema non prevede il rimpiazzo dei file originali con le loro copie grezze tantomeno l'eliminazione automatica di file vuoti (possibile risultato di un'istruzione `list`, `get` o `put` fallita)
4. Infine il sistema non prevede il trasferimento di file arbitrariamente grandi e la taglia massima consentiva è sull'ordine dei mega byte (per maggiori dettagli consultare `DESCRIPTION.md`)