Wielowątkowy serwer FTP oraz klient GUI

Wojciech Koszela 141251, Pavlo Ravliv 135412

1 Opis projektu

1.1 Zawartość

W skład projektu wchodzi:

1. `server/` (C) Wielowątkowy serwer FTP, zgodny z RFC 959 w zakresie podstawowych komend: ascii, binary, mkdir, rmdir, put, get. Wspiera również komendy: ls, system, abort, quit, polecenie PORT. Potwierdzone zostało jego poprawne działanie m.in. z klientem FTP GNU.
2. `client/` (Java) Prosty klient GUI pozwalający na komunikację z serwerem.

1.2 Struktura serwera

`main.c` - główny kod serwera, akceptujący połączenia od klienta i przetwarzający je w niezależnych sesjach w osobnych wątkach;
`ascii_read.h` - konwersja odczytywanych danych odpowiednio między formatem ASCII a binarnym i na odwrót; uwzględnia sytuacje przepełnienia bufora w wyniku konwersji,
`command.h` - parsowanie komend wysłanych przez klienta, analiza argumentów, dopasowywanie do wzorca,
`transfer.h` - obsługa przesyłania danych przez "data connection",
`session.h` - kontrola sesji użytkownika w oparciu o prostą maszynę stanową,
`utils.h` - funkcje pomocnicze.

1.3 Strukutra clienta

`Controller.java` - kod klienta odpowiadający za połączenie i obsługę strony przez którą chcemy podłączyć się do serwera
`MainGuiController.java` - klasa odpowiadająca za obługę przycisków głównego menu aplikacji
`Connect.fxml` - reprezantacja strony konekcyjnej w XMLu
`general-ui.fxml` - reprezentacja głównego menu klienta w XMLu
`ConnectionsMangar.java` - klasa reprezentująca połączenie
`Main.java` - punkt wejściowy do aplikacji

1.4 Sposób kompilacji i uruchomienia

Serwer:

```
$ cd server
$ make
$ ./ftp-server [-p port] [-r local_root]
```

Klient:
Należy pobrać JDK z JavaFX (link w repozytorium).
W IDE, np. Intellij IDEA zmienić odpowiednio na nie SDK projektu (client).
Uruchomić z główną klasą z Main.java.

2 Opis protokołu

Główna komunikacja klient-serwer odbywa się poprzez tzw. "control connection", natomiast do przesyłu danych, np. zawartości plików, otwierane jest osobne połączenie "data connection".

Control connection: dane transmitowane jak w protokole Telnet. Komunikaty zakończone są znakami powrotu karetki `<CR>` i nowej linii `<LF>`. Klient wysyła komendy składające się z maksymalnie 4-znakowego kodu komendy opcjonalnie z argumentem oddzielonym znakiem spacji, np. `LIST [<SP> dir] <CRLF>`. Serwer informuje klienta o stanie przetworzenia komendy, rozpoczętych operacjach, błędach za pośrednictwem komunikatów składających się z 3-cyfrowego kodu odpowiedzi i informacji dla użytkownika: `xyz <SP> info <CRLF>`.
`1yz` - pozytywna odpowiedź wstępna, `2yz` - pomyślne zakończenie, `3yz` - pozytywna odpowiedź pośrednia, `4yz` - przejściowe niepowodzenie, `5yz` - trwałe niepowodzenie.
Polecenia i odpowiadające im kody:
`abort` - `ABOR`,
`ascii` - `TYPE A`,
`binary` - `TYPE I`,
`ls` - `LIST [arg]`,
`mkdir` - `MKD dirname`,
`rmdir` - `RMD dirname`,
`put` - `STOR filename`,
`get` - `RETR filename`,
dodatkowo komendą `PORT a,b,c,d,e,f`, gdzie `a-f` to poszczególne bajty, informuje się serwer, że klient nasłuchuje na `a.b.c.d:ef`, czekając na utworzenie data connection.

Data connection: służy do przesyłania pozostałych danych, np. zawartości plików, czy wyniku polecenia `ls`. W przypadku ustawionego binarnego typu danych (image type) treść plików wysyłana jest w kolejnych 8-bitowych bajtach. Typ danych ASCII wymaga dodatkowo przekonwertowania każdego znaku nowej linii na sekwencję `<CRLF>`.