

# Dokumentacja projektu

Klient FTP

# Autorzy:

Poznań, 2016

# Spis treści:

1. Wstęp……………………………………………………………………………………………………………………………….…..2
2. Charakterystyka projektu……………………………………………………………………………………………..……...2
3. Czym jest protokół FTP………………………………………………………………………………………………………....3

3.1 Opis protokołu………………………………………………………………………………………………………………..3

3.2 Charakterystyka połączeń FTP……………………………………………………………………………………..….3

1. Opis kodu źródłowego…………………………………………………………………………………………………..…..….4

4.1 Opis protokołu FTP zastosowanego w programie……………………………………….…..…………..….4

4.2 Zastosowane funkcje w kodzie źródłowym………………………………………………………………………4

4.3 Opis komend programu……………………………………………………………………………………………………5

4.4 Zastosowane w komendach polecenia serwerowe oraz możliwe odpowiedzi serwera .….6

4.5 Przykład komunikacji klienta z serwerem…………………………………………………………………………7

1. Wymagania………………………………………………………………………………..…………………………………..……..8
2. Specyfikacja sprzętu………………………………………………………………………………………………………………8

## Wstęp

Niniejszy jest do dokumentacja projektu programistycznego utworzona na potrzeby zajęć pod tytułem Sieci komputerowe prowadzonych na Uniwersytecie im. Adama Mickiewicza w Poznaniu na wydziale Matematyki i Informatyki. W celu realizacji projektu posłużyliśmy się kodami o statusach serwera oraz zaawansowanym językiem programowania C.

## Charakterystyka projektu

Cele projektu jest utworzenie klienta serwerowego ftp do obsługi poleceń na danych typie serwera. Ma on ułatwić komunikację oraz przesyłanie danych. Dzięki zastosowaniu socketów można działać na portach, a komunikaty wykorzystane w programie pomogą w obsłudze jego. Natomiast dzięki nasłuchiwaniu odpowiedzi serwera zostaną wyświetlone komendy jakie serwer ukazał.

## Czym jest protokół FTP

### Opis protokołu

Protokół Transmisji Plików (File Transfer Protokol) jest obecny w Internecie od 1971 roku. Protokół ten wchodzi w skład zestawu TCP/IP. Jest to wygodne narzędzie, które pozwala przenosić dowolne pliki między dowolnymi komputerami w sieci Internet. Protokół ten pracuje poprawnie na dowolnym typie maszyny lokalnej, jak też i na dowolnej maszynie zdalnej, z którą to maszyną zdalną wymieniane są informacje. Programy FTP pozwalają zarówno sprowadzać pliki z sieci, jak też przekopiowywać pliki z lokalnego komputera na maszyny zdalne. Dostępne są przy tym opcje: wyświetlania katalogów, wyświetlanie plików w katalogach, pobierania informacji o bieżącym stanie pracy, ustawienia parametrów realizowanych zadań i operacji.

### 3.2 Charakterystyka połączeń FTP

Klient komunikuje się z serwerem za pomocą dwóch połączeń TCP

1. Sterujące, transmisja żądań i odpowiedzi:

– Serwer oczekuje na klienta na porcie 21.

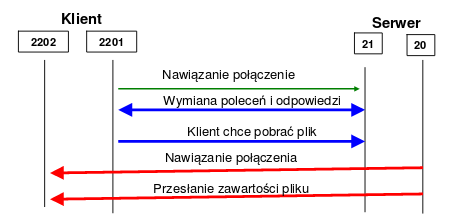
– Klient łączy się z dowolnego portu lokalnego do portu 21 serwera.

– Serwer pierwszy zamyka połączenie (na polecenie QUIT)

2. Danych, transmisja danych:

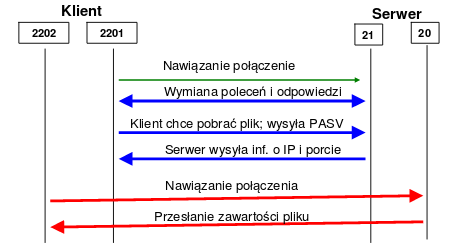
– Tryb aktywny:

1. Klient otwiera port i informuje o nim serwer (przekazuje IP i port)
2. Serwer łączy się do wskazanego portu i rozpoczyna się wymiana danych



– Tryb pasywny:

1. Serwer otwiera port i informuje o nim klienta (przekazuje IP i port)
2. Klient łączy się do wskazanego portu i rozpoczyna się wymiana danych



## Opis kodu źródłowego

### 4.1 Opis protokołu FTP zastosowanego w programie

Protokół FTP definiuje dwa oddzielne porty komunikacyjne. Na początku, po stronie klienta, inicjowane jest połączenia kontrolne, służące do przesyłania poleceń protokołu i odbioru odpowiedzi oraz kodów statusu operacji z serwera. Po wydaniu, poprzez połączenie kontrolne, odpowiednich poleceń następuje transfer danych, który wymaga utworzenia nowego połączenia. Połączenie przesyłu danych odbywa się za pomocą danych binarnych .   
W kolejnych punktach opisane zostaną typowe sekwencje poleceń. Funkcje modułu obsługi ftp mają nazwy odpowiadające poleceniom protokołu FTP, a zwracające kody, będą pokazywać statusy i kody błędów serwera FTP.

### 4.2 Zastosowane funkcje w kodzie źródłowym:

– Funkcja **clearString** służy do czyszczenia zmiennych typu string. Jako argumenty podaje się zmienną, a następnie ilość znaków do wyczyszczenia.

– Funkcja **hostname\_to\_ip** zamienia adres domeny na ip.

– Funkcja **main** zawiera poszczególne komendy potrzebne do poprawnej komunikacji z serwerem.

### 4.3 Opis komend programu

1. **ls** – wyświetla informacje o plikach w postaci listy.

Na początku serwer przechodzi w tryb pasywny dzięki komendzie serwerowej „PASV”. Następnie łączy się na ustalony port i wyświetla listę w wybranych typie kodowania.

1. **mkdir** – tworzy katalog o nazwie określonej przez użytkownika.
2. **rmdir** – usuwa katalog zdefiniowany przez użytkownika.
3. **cd** – służy do zmiany ścieżki dostępu aktualnego katalogu serwera.
4. **help** – wyświetla listę komend potrzebnych do obsługi programu.
5. **quit** – kończy połączenie z serwerem i zamyka program.
6. **pwd** – zwraca aktualną ścieżkę dostępu serwera.
7. **get** – służy do pobierania plików z serwera. Podobnie jak w przypadku ls ustanawia połączenie pasywne i serwer łączy się na wybrany port .
8. **put** – służy do przesyłania plików na serwer.
9. **close** – kończy połączenie z serwerem.
10. **rm** – usuwa plik zdefiniowany przez użytkownika.
11. **clear** – czyści okno programu.
12. **login** – ustanawiamy polaczenie z serwerem ftp wpisując nazwę host, port i dane potrzebne do zalogowania.

### 4.4 Zastosowane w komendach polecenia serwerowe oraz możliwe odpowiedzi serwera

1. PASV - przejdź w tryb pasywny i wystaw/odbierz dane na nowym porcie danych
   1. 227 Entering Passive Mode (ip1,ip2,ip3,ip4,porth,portl) -(w odpowiedzi podany jest adres serwera i numer portu bajt po bajcie adres ip to: ip1.ip2.ip3.ip4, port porth\*256+portl)
2. **TYPE** – wybór typu kodowania.
   1. 215 UNIX Type: L8 Remote system type is UNIX. Using binary mode to transfer files. - po wybraniu kodowania serwer komunikuje to.
3. LIST - wystaw na porcie danych zawartość aktualnego katalogu.
   1. 150 Opening mode data connection for '/bin/ls'. - po rozpoczęciu komunikacji serwer sygnalizuje.
   2. 226 Transfer complete - po ukończeniu transferu serwer sygnalizuje to.
4. MKD directory - utwórz katalog na serwerze
   1. 257 pathname was created - katalog directory został utworzony pomyślnie.
5. RMD directory - usuń katalog z serwera
   1. 250 RMD was successful - katalog directory został usunięty pomyślnie.
6. CWD new\_directory - Change Working Directory, zmień aktualny katalog
   1. 250 CWD was successful - aktualny katalog został zmieniony pomyślnie.
7. PWD - Print Working Directory, podaj aktualny katalog
   1. 257 „pathname” is current directory - aktualny katalog jest taki jak w cudzysłowie.
8. RETR file - wystaw na porcie danych nowe dane (zawartość pliku)
   1. 150 Opening binary mode data connection for 'file' - po rozpoczęciu komunikacji serwer sygnalizuje.
   2. 226 Transfer complete - po ukończeniu transferu serwer sygnalizuje to.
9. STOR file - odbierz na porcie danych nowe dane (zawartość pliku) i zapisz go na serwerze w aktualnym katalogu w pliku o nazwie file. Jeśli taki plik istnieje nadpisz go.
   1. 150 FILE: file - po rozpoczęciu komunikacji serwer sygnalizuje gotowość do odbioru danych.
   2. 226 Transfer complete - po ukończeniu transferu serwer sygnalizuje to.
10. DELE file - usuń plik z serwera
    1. 250 DELE was successful - plik file został usunięty pomyślnie.
11. USER user\_name - zaloguj nowego użytkownika o loginie user\_name,
    1. 331 Password required - w tym momencie serwer spodziewa się otrzymania zapytania PASS
    2. 503 Bad sequence of commands jeśli otrzyma inne zapytanie niż PASS po USER lub inne niż USER na rozpoczęcie komunikacji
12. PASS password - hasło logowanego użytkownika (to są dwa osobne zapytania, ale serwer spodziewa się zapytania PASS po USER)
    1. 230 User logged in

1. QUIT - wylogowanie użytkownika, zakończenie sesji.
   1. 221 Bye

### 4.5 Przykład komunikacji klienta z serwerem

* logowanie (kolejne podpunkty następują po sobie):

1. klient wysyła na port 21: USER michal
2. serwer wysyła na port 21: 331 Password required
3. klient wysyła na port 21: PASS haslo
4. serwer wysyła na port 21: 230 User logged in

* odbieranie pliku z serwera w trybie pasywnym:

1. klient wysyła na port 21: PASV
2. serwer wysyła na port 21: 227 Entering Passive Mode (127,0,0,1,4,1)
3. klient wysyła na port 21: RETR file.cpp
4. serwer wysyła na port 21: 150 Opening binary mode data connection for 'file.cpp'
5. serwer wysyła na port 1025: zawartość pliku file.cpp
6. serwer wysyła na port 21: 226 Transfer complete

* Ad. odbieranie pliku:

przed wysłaniem odpowiedzi 227 Entering Passive Mode (127,0,0,1,4,1) - serwer powinien zacząć nasłuchiwać na porcie, który wysyła w odpowiedzi, dopiero jak to się uda wysyła odpowiedź, w naszym przypadku adres ip, będzie zawsze ten sam, co dla komunikacji na porcie 21.

wybrany port to 1025 (4\*256+1), dlatego zawartość pliku file.cpp jest wysyłana na tym porcie.

## Wymagania

Program jest pisany w języku programistycznym C. Działa on pod platformą systemów typu UNIX wykorzystując biblioteki wbudowane. Program jest w formie tekstowej aczkolwiek przejrzysty ułatwiający obsługę. Zarówno osoba starsza, jak i młodsza znająca podstawy języka angielskiego poradzi sobie z obsługa danego programu.

## Specyfikacja sprzętu

- system operacyjny typu Unix

- procesor: Intel Pentium bądź lepszy

- karta graficzna: zintegrowana

- pamięć: 256 MB pamięci RAM

-30 MB wolnego miejsca na dysku twardym

- dostęp do internetu