Operating Systems and Networks (159.342)

Assignment 1: FTP server (Cross-platform, Active mode, IPv6-compliant), due 11th of April

Requirements

Your main task is to write a simple Cross-platform, Active mode FTP *server* program using the socket API (preinstalled in the operating system: Windows/Linux/MacOS). The FTP server must be able to process the following commands from the built-in FTP user agent: USER, PASS, OPTS, TYPE, LIST, EPRT, PORT, RETR and QUIT. Your FTP server program must satisfy the following requirements:

1. It should be RFC 959 protocol-, RFC 2428 protocol- and IPv6-compliant.
2. It should not use data structures that work only on IPv4 addresses (penalty applies).
3. It must compile using the latest g++ 14.2.0 compiler (see download link from our Stream website) or g++ 8.1 (installed on Massey's lab machines).
4. It must be cross-platform but will be tested only on Windows 11.
5. It should be able to accept an optional ephemeral port to listen to, as one of its arguments (e.g. server 1122)
6. If the port number is not specified, it should use port 1234 as default.

The FTP server will be tested using the built-in FTP client in Windows 11 (IPv6), using simple commands such as:

- dir
- binary
- ascii
- get <filename>

For testing purposes, include one authorized user entry in your server with the user name: *napoleon*, and password: *342*

Guidelines

Learn about the protocol commands and response messages that need to be issued. Refer to materials provided in the stream web site (i.e. RFC 959, RFC 2428, lecture slides, sample codes). It is also highly recommended that you try to interact with a real ftp server (e.g. Internet Information Services (IIS) - ftp server), to see how it actually responds to your requests.

When the FTP server receives an OPTS command, simply return 502 unrecognized command.

Remember that the FTP protocol requires two parallel TCP connections: one in port 21 and a data connection in port 20. The machines in the lab may not allow users to bind applications on these ports. Therefore, your program should allow for any valid ephemeral port i.e. [1024, 65535] as an alternative port number.

Marking the assignment:

The assignment is going to be marked based on functionality. The server should not crash if the user tries to issue commands that are not implemented. The marks are distributed as follows:

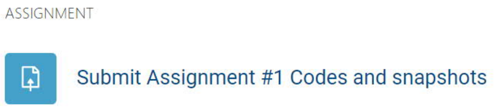
- 2 marks for correct connection/disconnection and resource allocation and de-allocation (the server should allow for reconnection after errors, etc.)
- 2 marks for the EPRT command, including error handling in IPv6
- 2 marks for LIST command, including error handling, in IPv6
- 2 marks for the RETR command (get <filename>), including error handling in IPv6. Your program should work with binary files (e.g. jpg files)
- 1 mark for TYPE <mode> command; mode is either 'I' (for image), or 'A' (for ASCII). To test the image/binary mode, try transferring a jpg image file from your FTP server.
- 1 mark for the brief documentation and accomplished checklist
- This assignment is worth 10 marks. Plagiarized works will be penalized.

Submission Instructions:

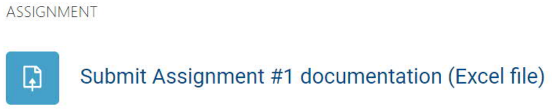
1. Submit your complete C/C++ program electronically via Stream.
2. Use the following filename for your FTP server: server.cpp
3. Submit the makefile for building your codes. The makefile should suffice to build your codes.
4. As a backup, if you developed your program using an IDE, such as VSCODE or Clion, also submit the complete

support files (e.g. CMakeLists.txt, project file, *.sln, *.JSON, etc.), for building your codes.

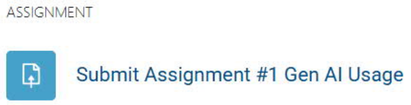
5. Assignment documentation file: Download and fill out the Excel file, named assignment_documentation.xlsx to provide important information about your work. Rename the file using your ID_number (e.g. 87654321.xlsx). This will help facilitate accurate marking of your assignment. Deductions will be made if you do not submit the completed Excel file.
6. Snapshots file: Create a document containing snap shots of sample FTP server-client interactions demonstrating EACH of the required commands. Label and briefly describe each image. You may create a Word document or a pdf file to accomplish this task.
7. Create a zip file containing the following files:
   - Complete set of codes (including *.cpp, *.h, makefile, other supporting files)
   - Snapshots file
8. Submit the zip file containing your complete program and snapshots electronically via Stream. You will find a button with the following label.



9. Submit the assignment documentation file via Stream website. You will find a separate button with the following label.



10. Lastly, submit the Gen AI Usage documentation file via Stream website. You will find a separate button with the following label.


ASSIGNMENT
Submit Assignment #1 Gen AI Usage