Assignment 3
CS-471
Spring 2013

Due Date: 5/26/2013 at 11:59 pm (No extensions) You may work in groups of 6

Goals:

1. To further acquainted yourself with network applications development.
2. Learn how to resolve domain names to IP addresses.
3. To implement an FTP server and client.
4. To learn about the latest features of the Linux kernel meant to simplify network programming.
5. To learn the challenges of protocol design.
6. To appreciate the challenges of developing complex network applications.

Overview

In this assignment, you will implement both (simplified) FTP server and client. The client shall connect to the server and be able to upload and download files to/from server.

Specifications

The server shall be invoked as:

```
./serv <PORT NUMBER>
```

`<PORT NUMBER>` specifies the port at which ftp server accepts connection requests.

For example: `./serv 1234`

The ftp client is invoked as: `cli <server_machine> <server_port>`
`<server_machine>` is the domain name of the server (`ecs.fullerton.edu`). This should be converted to corresponding 32 bit IP address using the `getaddrinfo()`. http://www.beej.us/guide/bgnet/output/html/multipage/getaddrinfoman.html.
For example: `./cli ecs.fullerton.edu 1234`

Upon connecting to the server, the client prints out **ftp>**, which allows the user to execute the following commands.

`ftp> get <file name>` (downloads file <file name> from the server)
`ftp> put <filename>` (uploads file <file name> to the server)
`ftp> ls` (lists files on the server)

`ftp> quit` (disconnects from the server and exits)

Use two connections for each ftp session - control and data. Control channel lasts throughout the ftp session and is used to transfer all commands (ls, get, and put) from client to server and all status/error messages from server to client. The initial channel on which the client connects to server will be the control channel. Data channel is used for data transfer. It is established and torn down for every file transfer – when the client wants to transfer data (ls, get, or put), it generates an ephemeral port to use for connection (use `getsockname()` http://man7.org/linux/man-pages/man2/getsockname.2.html). The client shall then send the ephemeral port number to the server, and then wait for the server to connect to the client on that port. The connection is then used to upload/download file to/from the server, and is torn down after the transfer is complete.

For each command/request sent from the client to server, the server prints out the message indicating SUCCESS/FAILURE of the command. At the end of each transfer, client prints the filename and number of bytes transferred.

Designing the Protocol

Before you start coding, please design an application-layer protocol that meets the above specifications. Please submit the design along with your code. Here are some guidelines to help you get started:

- What kinds of messages will be exchanged across the control channel?
- How should the other side respond to the messages?
- What sizes/formats will the messages have?
- What message exchanges have to take place in order to setup a file transfer channel?
- How will the receiving side know when to start/stop receiving the file?
- How to avoid overflowing TCP buffers?
- You may want to use diagrams to model your protocol.

Tips

- All sample files are in directory `sample`. Type `make` to build all programs.
- Please see sample file, `ephemeral.cpp`, illustrating how to generate an ephemeral port number. This program basically creates a socket and calls `bind()` in order to bind the socket to port 0; calling `bind()` with port 0 binds the socket to the first available port.
- Please see sample files, `getaddrinfo.cpp`, illustrating how to convert the host name into an IP address using `getaddrinfo()` function.
- You may use `popen()` system call in order to get a list of files in the directory. Please see sample file, `popen.cpp`, illustrating the usage.

- You are highly encouraged to use the `sendfile()` system call in order to send files. This is a recent addition to the Linux kernel that was meant to improve performance of FTP and web servers. Please see `sendfile` directory for sample codes. The directory basically features the server and client from quiz 3, with one difference: client uses `sendfile()` system call in order to transfer the file. Also, see http://linuxgazette.net/issue91/tranter.html for more information on using `sendfile()`.

EXTRA CREDIT:

Implement the above using a forking server and a threaded server.

SUBMISSION GUIDELINES:

- This assignment may be completed using C++, Java, or Python.
- Please hand in your source code electronically (do not submit .o or executable code) through **TITANIUM**. You must make sure that this code compiles and runs correctly.
- **Only one person within each group should submit.**
- Write a README file (text file, do not submit a .doc file) which contains
  - Names and email addresses of all partners.
  - The programming language you use (e.g. C++, Java, or Python)
  - How to execute your program.
  - Whether you implemented the extra credit.
  - Anything special about your submission that we should take note of.
- Place all your files under one directory with a unique name (such as `p1-[userid]` for assignment 1, e.g. `p1-mgofman1`).
- Tar the contents of this directory using the following command. `tar cvf [directory_name].tar [directory_name]` E.g. `tar -cvf p1-mgofman1.tar p1-mgofman1/`
- Use TITANIUM to upload the tared file you created above.

Grading guideline:

- Protocol design 5'
- Program compiles: 5'
- Correct get command: 25'
- Correct put command: 25'
- Correct ls command: 10'
- Correct format: 10'

- Correct use of the two connections: 15'
- README file included: 5'
- BONUS: 15 points
- Late submissions shall be penalized 10%. No assignments shall be accepted after 24 hours.

Academic Honesty:

Academic Honesty: All forms of cheating shall be treated with utmost seriousness. You may discuss the problems with other students, however, you must write your **OWN codes and solutions**. Discussing solutions to the problem is **NOT** acceptable (unless specified otherwise). Copying an assignment from another student or allowing another student to copy your work **may lead to an automatic F for this course**. Moss shall be used to detect plagiarism in programming assignments. If you have any questions about whether an act of collaboration may be treated as academic dishonesty, please consult the instructor before you collaborate. Details posted at http://www.fullerton.edu/senate/documents/PDF/300/UPS300-021.pdf.