1. Implement a ***concurrent*** server and a client using TCP socket for the following functionality.

1. The client accepts through the command line..

a) Server's IP address and port number

b) A name of a text file (say input\_data)

c) A name of an output file (explained in point 5 below)

(The file "input\_data" consists a list of strings (separated by newline) which are assumed to be file or directory names in the file system of the server)

2. The client sends the content of the file "input\_data" to the server.

3. The server process, after accepting the connection from the client process, has to receive these.

4. The server finds whether the strings are “***regular files***" or "***directory files***".

(Hint: Use the stat system call for this purpose).

(For a regular file, the server sends back the filename followed by the size of the file. For a directory the server sends back the directory name along with the string "DIRECTORY")

(If any of the file/directory is not found in the present working directory of the server, the server sends back "ERROR" for that case)

(Consider space to be a delimiter between a filename and the file size)

5. The client saves the result in a file say "output\_data", which was received as the second command line argument. Each line will be having the required data.

Like say

file\_1 300

file\_2 DIRECTORY

file-3 400

file\_4 ERROR

Where the initial "input\_data" contained,

file\_1

file\_2

file\_3

file\_4

6. There should be separate functions for sending data/receiving data/processing data etc

2) You need to write a client and a server utilizing TCP socket connections. The following are the requirements.

a) The client must be able to connect to the server and obtain a list of available clients.

b) After obtaining this list, the client must be able to connect to one other client on the list.

c) The clients must then be capable of relaying text messages directly to one another

Without further support from the server.

(You can define whatever convention you would like to indicate that a text message is complete (e.g.hitting [Enter],etc).

d) Each client must send an acknowledgement to the other client that a text message has been received and displayed.

e) You must also provide some way to disconnect and end the conversation.

(After disconnecting, the user should be able to retrieve a new copy of the server’s

Client list and connect to a new client.)

f) The connection between the two clients must be duplex, so both clients are capable of transmitting and receiving at the same time.

1. The basic requirements of a webserver is given below. You need to implement this.

|  |  |  |
| --- | --- | --- |
| Web Server Implementation  (The scope of the project is limited to building an application program (it acts as the server) that accepts connections in order to service requests. It then parses the request and sends back responses. The user agent can be a standard browser or a commnad line tool like curl (http://en.wikipedia.org/wiki/CURL) can be used.) | | |
| Requirement Tag | Requirement Description | Comments |
| Functional Requirements |  |  |
| WSS\_01 | The implementation will follow Hypertext Transfer Protocol -- HTTP/1.1 | Mandatory |
| WSS\_02 | The application should be able to parse the request and generate logs, which clearly identifies the Request Line and the message body. Further the Request line also needs to be parsed into different headers and their values. | Mandatory (Ref. Page- 35 of pdf document of RFC 2616) |
| WSS\_03 | The application should be able to provide the response ( Status Line and the message body), based on the request. A subset of status code can be chosen. | Mandatory |
| WSS\_04 | The response should also be logged. | Mandatory |
| WSS\_05 | GET, HEAD, PUT, POST methods need to be implemented. | Mandatory |
| WSS\_06 | TCP "persistent connection" for a request needs to be implemented. | Mandatory |
| WSS\_07 | The server should preferably be multithreaded. | Desirable |