Project Specifications - I

- **Objective: implement a "simple FTP server", i.e. a server that lets clients read and write files from its own (remote server) disk**
- Connection should be stateful: the client connects to the server, sends a set of requests and closes the connection (hint: use stream sockets)
- The server should be able to manage an unlimited number of clients, with minimal delay to establish a connection (hint: use `fork()` to create a dedicated process server for each client)
- The shared portion of the server file system is one and the same for all clients

Project Specifications - II

- The client executable should accept the following option on the command line:
  - IP address or DNS name of the server
- When started, the client should immediately connect to the server on a well-known port
- Supported client commands: `ls`/`get`/`put` (same syntax as SFTP, no options – see `man sftp`) using only relative pathnames on the server and the client
- The client should wait for a new command from its standard input and send it immediately to the server
- The client should loop forever (i.e., till it reads an `exit` command)

Project Specifications - III

- The server should correctly manage file ownership and access rights, i.e.:
  - require client authentication with user and password, as defined in the server
    - password encryption not required
  - file access should be subject to client rights for all operations (read, write, create, delete)
    - hint: use `setuid()`
  - root access should not be allowed
  - define a “home” directory for each user

Project Specifications - IV

- The server executable should wait for input on a well-known port
- The server should shutdown when it receives a user-selected signal or a `quit` command from the command line.
- The server executable should accept the following options on the command line (default values should be stored in a configuration file):
  - root directory of the shared portion of file system