Exercise 1: Multi-Client TCP Server

A non-blocking TCP server and interactive client for a Communication Networks course assignment.

▶ Table of Contents

About the Project

- A small non-blocking TCP server and interactive client for a networks exercise.
- The server uses `select()` for I/O multiplexing and supports multiple simultaneous clients.
- Implements a two-step login (username, then password) and several commands: balanced parentheses checking, LCM, and Caesar cipher.

(back to top)

Repository Files

- `ex1_server.py` — Non-blocking TCP server using `select()`. Loads users from a tab-delimited file and manages per-client authentication and command state.
- `ex1_client.py` — Interactive TCP client. Connects to the server, handles login, and sends commands.
- `users_list.txt` — Tab-delimited user credentials file.
- `README.md` — This file.

(back to top)

Quick Usage

1. **Start the server:**

   ```
   python ex1_server.py users_list.txt [port]
   ```

   - `users_list.txt`: Tab-delimited file, each line is `username<TAB>password`.
   - `port`: Optional, default is 1337.

2. **Start the client:**

   ```
   python ex1_client.py [host] [port]
   ```

   - `host`: Default is `localhost`.
   - `port`: Default is `1337`.

(back to top)

Protocol

Authentication

1. Server sends: `Welcome! Please log in.`
2. Client sends: `User: <username>`
3. Server replies: `OK`
4. Client sends: `Password: <password>`
5. Server replies:
   - Success: `Hi <username>, good to see you`
   - Failure: `Failed to login.` (client can retry)

Commands (after login)

- `parentheses: <text>` — Checks if `<text>` contains only '(' and ')' and is balanced.

  Replies: `the parentheses are balanced: yes|no` or `ERROR: The string isn't only parentheses`
- `lcm: <X> <Y>` — Computes least common multiple of two integers.

  Replies: `the lcm is: <result>` or error message.
- `caesar: <text> <shift>` — Caesar cipher encryption.

  `<text>`: English letters and spaces only.

  Replies: `The ciphertext is: <result>` (lowercase) or `error: invalid input`
- `quit` — Closes the connection.
- Unknown or malformed commands:

  Replies: `ERROR: ...` and closes the connection.

(back to top)

Design Decisions

- **TCP** is used for reliable, ordered, connection-oriented communication.
- **select()** enables non-blocking, multi-client handling in a single-threaded server.
- **Explicit two-step login**: Username and password are sent in separate messages for clarity and protocol simplicity.
- **Strict parsing**: Commands must match the expected format; invalid commands result in disconnection.
- **Per-client state**: The server tracks each client's authentication and command state.

(back to top)

Notes

- The users file **must** use tabs (`\t`) between username and password.
- The client expects one-line user input for each prompt.
- If you see "not enough values to unpack" or login issues, check your users file for correct formatting.
- The server and client require **Python 3.10+** (for `match`/`case`).

(back to top)

Authors

- Guy Harem
- Karen Goldberg