Exercise 1 – Client–Server Program

Computer Networks – Fall Semester 2025

Part 1 (85%) – Client-Server Program

In this exercise, you will write a server program that allows clients to use several operations performed on the server. You will define an application-layer protocol using either **TCP** or **UDP**, according to your choice.

Exercise Goals

- Proper use of socket programming in network environments.
- Defining and developing an application-layer communication protocol.
- Supporting multiple clients simultaneously.

Server

Run the server with:

```
./ex1_server.py users_file [port]
```

Parameters

`users_file` A path to a simple **tab-delimited text file** containing two fields:

```
username<TAB>password
```

Example:

```
Bob     simplepass
Alice   BetT3RpAas
```

There must be exactly one tab between fields.
You can asssume clients cannot be added while the server is running.

`port` Optional.
Default: **1337**

Server Behavior

The server: - runs continuously
- does not exit on its own
- serves multiple clients at the same time
- uses `select` (not multi-threading)
- accepts new connections at any moment

Client

Run the client with:

```
./ex1_client.py [hostname [port]]
```

Defaults:

```
hostname = localhost
port = 1337
```

You may specify hostname without port, but not port without hostname.

`hostname` may be a name or an IP address.

Login Procedure

When a client connects, the server sends:

```
Welcome! Please log in.
```

The server expects:

```
User: username
Password: password
```

If valid:

```
Hi {username}, good to see you
```

If invalid:

```
Failed to login.
```

and then will be given the opportunity to log in again.

Supported Commands

1. Parentheses Balance Check

`parentheses: X`

Where `X` is a sequence of parentheses.

Example:

`(()()(`

Server responds:

`the parentheses are balanced: yes/no`

---

2. LCM Calculation

`lcm: X Y`

Where `X` and `Y` are signed integers.

Server responds:

`the lcm is: R`

---

3. Caesar Cipher

`caesar: plaintext X`

Where: - `plaintext` is a character string
- `X` is an integer shift

Server returns:

`the ciphertext is: Y`

Rules: - apply shift to English **letters only**
- any non-alphabetic character produces:

`error: invalid input`

---

4. Quit

`quit`

After finishing the command, the server closes the connection.

Error Handling

Malformed commands or incorrect formats must produce an appropriate error message.

Exercise Requirements

You must implement both server and client exactly as described.
Choose TCP or UDP and document your design in a **Readme** file.

Your code must: - use the socket API properly
- include correct error handling
- match the protocol you define

Part 2 (15%) – Using Wireshark

Download Wireshark:

https://www.wireshark.org

Steps: 1. Run Wireshark.
2. Choose the correct network interface.
3. Apply a relevant filter.
4. Capture packets exchanged between client and server: - connection
- commands
- responses

Save the capture as a `.pcap` file.

You must also: - choose a portion of the communication process - include a screenshot from wireshark - add a short explanation describing what the protocol is doing at that moment

---

Submission Instructions

Submit a single ZIP file named:

```
EX1_ID1_ID2     (pair)
EX1_ID          (individual)
```

ZIP must include: - all source code
- any modules you added
- all Python files starting with:

`#!/usr/bin/python3`

- `Readme.pdf` describing your protocol
- `.pcap` file + screenshot + explanation

Example Interaction

Server:

```
./ex1_server.py ~/my_dir/users_file.txt
```

Client 1:

```
./ex1_client.py
Welcome! Please log in.
User: Bob
Password: simplepass
Hi Bob, good to see you
caesar: Hello! 2
error: invalid input
```

Client 2:

```
./ex1_client.py
Welcome! Please log in.
User: Alice
Password: BetT3RpAas
Hi Alice, good to see you
parentheses: ((()))()(())
the parentheses are balanced: yes
lcm: 6 21
the lcm is: 42
quit
```

Client 1 continues:

```
caesar: Hello 2
the ciphertext is: jgnnq
quit
```

Good luck!

Computer Networks – Fall Semester 2025