Week 14: Practicum 2

CS5600 Spring, April 2025
Shorena K. Anzhilov

This project comprises two C programs: `client.c` and `server.c`. Together, they implement a client-server architecture over TCP sockets, enabling file operations such as uploading (`WRITE`), downloading (`GET`), deleting (`RM`), and listing (`LS`) files.

Key features include:

- ➢ Multi-threaded Server: Allows concurrent client connections.
- ➢ File Versioning: Maintains multiple versions of uploaded files.
- ➢ XOR-based Encryption/Decryption: Secures file transfers.
- ➢ Graceful Shutdown: Handles `SIGINT` for safe server termination.

`client.c`

Lines 1–20:

- ☐ **Purpose:** Include necessary headers and define constants.
- ☐ **Key Feature:** Defines `ENCRYPTION_KEY` for XOR cipher.

Lines 22–27:

- ☐ Function: `xor_cipher`
- ☐ Purpose: Implements XOR encryption/decryption.
- ☐ Feature: Encryption/Decryption

Lines 29–31:

- ☐ **Function:** `main`

- ☐ **Purpose:** Entry point; checks for sufficient arguments.

Lines 33–43:

- ☐ **Purpose:** Establishes TCP connection to the server.
- ☐ **Feature:** Network Communication

Lines 45–78:

- ☐ **Command:** `WRITE`
- ☐ **Purpose:** Encrypts and uploads a file to the server.
- ☐ **Features:** Encryption, File Upload

Lines 80–129:

- ☐ **Command:** `GET`
- ☐ **Purpose:** Downloads and decrypts a file from the server.
- ☐ **Features:** Decryption, File Download, Versioning

Lines 131–142:

- ☐ **Command:** `RM`
- ☐ **Purpose:** Sends a delete request to the server.
- ☐ **Feature:** File Deletion

Lines 144–157:

- ☐ **Command:** `LS`
- ☐ **Purpose:** Requests a list of files from the server.

- ☐ **Feature:** File Listing

Lines 159–161:

- ☐ **Purpose:** Handles invalid commands.

Lines 163–164:

- ☐ **Purpose:** Closes the socket and exits.

`server.c`

Lines 1–20:

- ☐ **Purpose:** Include necessary headers and define constants.
- ☐ **Key Feature:** Defines `ENCRYPTION_KEY` for XOR cipher.

Lines 22–27:

- ☐ **Function:** `xor_cipher`
- ☐ **Purpose:** Implements XOR encryption/decryption.
- ☐ **Feature:** Encryption/Decryption

Lines 29–36:

- ☐ **Function:** `make_parent_dirs`
- ☐ **Purpose:** Creates necessary directories for file storage.
- ☐ **Feature:** File Management

Lines 38–49:

- ☐ **Function:** `get_latest_version`

- ☐ **Purpose:** Determines the latest version number of a file.
- ☐ **Feature:** File Versioning

Lines 51–60:

- ☐ **Function:** `list_files`
- ☐ **Purpose:** Sends a list of files to the client.
- ☐ **Feature:** File Listing

Lines 62–66:

- ☐ **Function:** `handle_sigint`
- ☐ **Purpose:** Handles `SIGINT` for graceful shutdown.
- ☐ **Feature:** Signal Handling

Lines 68–171:

- ☐ **Function:** `handle_client`
- ☐ **Purpose:** Processes client requests in a separate thread.
- ☐ **Features:**

Lines 73–113: `WRITE` command handling.

- ☐ **Feature:** File Upload, Versioning

Lines 115–153: `GET` command handling.

- ☐ **Feature:** File Download, Versioning

Lines 155–164: `RM` command handling.

- ☐ **Feature:** File Deletion

Lines 166–170: `LS` command handling.

- ☐ **Feature:** File Listing

Lines 173–192:

- ☐ **Function:** `main`
- ☐ **Purpose:** Sets up the server socket, binds, listens, and accepts connections.
- ☐ **Features:**

Line 175: Sets up `SIGINT` handler.

Lines 180–191: Accepts client connections and creates threads.

- ☐ **Feature:** Multi-threading

Features Tested:

- WRITE (Part 1)
- GET (Part 2)
- RM (Part 3)
- Versioning (4b)
- Multi-client (4a)
- Encryption (4c)
- LS (4d)
- SIGINT (4d)

Part 1: WRITE (Encrypted File Upload)

- ☐ make

- [ ] ./server

Then I opened a new terminal and ran:

- [ ] echo 'Version 1 content' > test.txt   I used Single quotes only, “ ” double did not work.
- [ ] mkdir server_storage
- [ ] ./client WRITE test.txt myfile.txt

- [ ] echo 'Version 2 content' > test.txt
- [ ] ./client WRITE test.txt myfile.txt

- [ ] echo 'Version 3 content' > test.txt
- [ ] ./client WRITE test.txt myfile.txt

To check the files I ran :

- [ ] ls server_storage

Expected output:

- ❖ myfile_v1.txt myfile_v2.txt myfile_v3.txt

Part 2: GET (Decryption + Versioning)

- [ ] ./client GET myfile.txt latest.txt
- [ ] cat latest.txt

It showed me decrypted latest version → “ Version 3 content”

- [ ] ./client GET myfile.txt:1 v1.txt
- [ ] cat v1.txt

It showed me → “Version 1 content”

- [ ] ./client GET myfile.txt:2 v2.txt
- [ ] cat v2.txt

It showed me → “ Version 2 content”

Part 3: RM (Remote Delete)

- [ ] ./client RM myfile_v1.txt
- [ ] ls server_storage

`myfile_v1.txt` was deleted.

Part 4b: Versioning

Versioning has already been demonstrated by:

- ☐ Writing multiple versions
- ☐ Retrieving specific versions using `GETversion`
- ☐ Viewing versioned files in `server_storage`

Part 4d: LS (List Server Files)

- ☐ ./client LS
- ☐ ./client LS myfile

myfile_v2.txt

myfile_v3.txt

__END__

Part 4a: Multi-client (Threaded Server)

I Opened **two terminals**, ran the following at the same time:

Terminal 1:

- ☐ ./client GET myfile.txt:2 testA.txt
- ☐ ./client GET myfile.txt:3 testD.txt

Terminal 2:

- ☐ ./client GET myfile.txt:2 testB.txt
- ☐ ./client GET myfile.txt:3 testD.txt

After GET's were completed:

- ☐ cat testA.txt

It showed "Version 2 content"

- ☐ cat testB.txt

It showed "Version 2 content"

- ☐ cat testD.txt

It showed "Version 3 content:

Part 4c: Encryption

To check the encrypted file directly I ran:

- ☐ cat server_storage/myfile_v2.txt

The output was unreadable binary data.

- ☐ %KSs%

To Check the decrypted file:

- ☐ cat v2.txt

It showed “Version 2 content”

Part 4d: SIGINT Handling (Safe Shutdown)

While the server is running, I pressed:

- ☐ Ctrl + C

And I got an output:

- ☐ Caught SIGINT, closing server socket...