

CS3205 (Semester: Holi 2025) Programming Assignment 2

Submission Deadline: *Friday, 28th March, 2025*

Total:

40 Marks

Exercise 1. [15 marks]. Develop a networked directory synchronization application that operates between a central server and multiple networked clients. The application consists of two components: (a) a server component and, (b) a client component. The server monitors a designated “sync” directory on the local host and tracks file creation and deletion events, including subdirectories. These updates are communicated to all connected clients. Each client maintains a local directory that mirrors the server’s sync directory, excluding specific file types.

Both the client and server components must be implemented using multithreading. Use TCP communication. To detect changes in the server’s sync directory, use the Linux’s [inotify](#) API (`<sys/inotify.h>`). Before integrating network communication, study and understand the `inotify` API, including the relevant system calls and flags.

The server is run as:

```
$/syncserver path_to_local_directory port max_clients
```

The client is run as:

```
$/syncclient path_to_local_directory path_to_ignore_list_file
```

Server component specification:

1. The server utilizes the `inotify` API to continuously monitor the specified **sync directory**.
2. The directory-watching function must operate recursively, ensuring that all newly created subdirectories are also tracked.
3. The server does **not** track file modifications, only **file and directory creation, deletion, and movement**.
4. Use the following inotify flags when adding watches:
`IN_CREATE | IN_DELETE | IN_MOVED_FROM | IN_MOVED_TO`
5. The server simultaneously supports a maximum number of clients (`max_clients`), as specified in the command-line argument.
6. When a client connects, the server:
 - Sends updates about changes in the sync directory, except those matching the client’s ignore list.
 - Transfers any newly created files to the client, ensuring that the client’s local sync directory remains up-to-date.
 - Each client is tracked based on its IP address.
7. The server maintains the ignore list (see client component) for each client in memory and removes it when the client disconnects.

Client component specification:

1. The **ignore list file** specifies file extensions that the client does **not** want to sync. It contains a **comma-separated list of extensions** (e.g., .mp4,.exe,.zip).
2. Upon connecting to the server, the client sends its ignore list to the server.
3. The format for sending this ignore list is **user-defined**—students must document their chosen approach within their code.

Optional (marks will not be awarded for this in this assignment): Check for modifications in the watched list, and in case of modification send the new file to the client, who should overwrite the existing file.

Note: Use only the terminal to move, copy or delete files in the sync directory.

Exercise 2. [20 marks]. In this exercise, you will implement a two-player ping pong game that operates over a local area network (LAN) using TCP sockets. The game consists of a server and a client communicating in real-time to maintain a synchronized game state.

Game Structure:

- One player acts as the server, waiting for a client to join.
- The second player acts as the client, connecting to the server to initiate the game.
- Players control paddles that can move left and right to bounce the ball toward their opponent.
- If a player misses the ball, their opponent scores a point.
- The game state, which includes paddle positions, ball position, scores, and other necessary elements, is exchanged between the server and client at regular intervals (few milliseconds).


The server is run as:

```
$/pingpong server PORT
```

The client is run as:

```
$/pingpong client <serverip>
```

A boilerplate code is provided to you that implements a basic GUI with key capture that can move the game paddle in the left or right direction. A ball is simulated to move over the game's canvas that bounces off the paddle. [Link to code: [pingpong.c](#)] [Link to code output preview:

 demo.mp4]

The current code uses a “paddle” - in your implementation you need to have two paddles, viz., **paddleA** (**paddleA** can be the server by convention) and **paddleB**. Also you need to update the **update_paddle(char)** function to incorporate sending out the local updates to the opponent and receiving the opponent's updated state to reflect its respective paddle position.

Interactive speeds [5 marks]. Out of total 20 marks, 5 marks are reserved for interactive speeds between the two players that result in a smooth user experience for a sustained period of time.

Exercise 3 [5 marks]. We will update a Google form link here to take feedback on the use of GenAI for this assignment. You need to fill up this form reflecting the efforts spent on the above two assignments. It is okay if you have generated the entire code using AI tools but it is required that you understand how to debug or update the code. We will not penalize you for AI generated code. After you submit the form, save it as a PDF file. The PDF file needs to be submitted.

https://docs.google.com/forms/d/e/1FAIpQLSd29qOEqwwbMREIO6T8lfn-kD_yxEKfRuotKZkYEe8iFKI5g/viewform?usp=preview

Submission Instructions: Create a folder called CS3205_Assignment2_roll1_roll2. Have three subfolders: ex1, ex2 and ex3. In each folder put the relevant fully documented source code. Compress the root folder as a gzipped file and submit CS3205_Assignment1_roll1_roll2.zip.

One person per group should submit the file in Moodle.

Late Submission. The deadline for submitting this assignment is 11:00 pm, 28th March. After this there will be a late penalty of 25% for each day for the next four days. You are not required to submit the assignment solutions beyond 31st March.

Viva. We will conduct a viva for every group. The viva will not be limited to the specific questions asked in this assignment, but to the specific topics covered here.