Multiplayer Online Quiz Game

Description:

This project entails the development of a multiplayer online quiz game, allowing simultaneous participation of multiple players. Players engage in various quiz categories, answering questions within a limited time frame to compete for scores. The system ensures real-time updates, exception handling, dynamic memory management, process synchronization, concurrency, and networking capabilities. The implementation will be in C within the Linux environment.

Implementation Overview:

1. Quiz Management:

You are tasked with developing a comprehensive management system for the multiplayer online quiz game. Your objective is to design a sophisticated data structure to represent quizzes, encompassing essential details like quiz categories, questions, answers, and scores. Implementing functions to initiate new quizzes, manage active quizzes, and facilitate the quiz process is crucial. Additionally, you must devise a system for handling quiz closures once the time limit for answering questions expires.

Key functionalities to implement:

- **Creating New Quizzes:** Develop functionality to start fresh quiz sessions across various categories, enabling a diverse range of topics and themes for participants. Make sure the quizzes are randomized.
- **Real-Time Update of Quiz Status:** Ensure timely updates of ongoing quiz sessions, providing participants with up-to-date information on active categories and questions to enhance engagement and participation.
- **Managing the Quiz Process:** Oversee the entire quiz process, including validating participant answers to maintain fairness and accuracy. Implement mechanisms to manage user responses, validate correctness, and update scores accordingly.
- **Closing Quizzes:** Implement a systematic approach to gracefully conclude quiz sessions once the allotted time for answering questions elapses. Notify all participants appropriately and finalize scores to provide a conclusive end to the quiz.
- **Quiz History:** Maintain comprehensive records of quiz sessions, including participant scores, questions attempted, and correct answers. This historical data serves to enhance transparency and provides valuable insights for future quiz management.

Moreover, the multiplayer online quiz game should support concurrent quiz sessions to accommodate multiple participants simultaneously. Implementing a robust communication protocol is essential to efficiently manage multiple concurrent clients and facilitate seamless interaction between clients and the server. Emphasis should be placed on optimizing

communication efficiency, ensuring data integrity, and providing a smooth user experience for all participants.

2. User Interface:

Your task is to design a user-friendly interface that empowers players to seamlessly interact with the multiplayer online quiz game. Develop views that facilitate the following functionalities:

- **Join the game:** Implement registration and login features to allow players to enter the game environment securely and efficiently.
- **Select quiz categories:** Enable players to explore and choose from a wide range of quiz categories like science, history, sports, and more, ensuring diverse and engaging gameplay experiences.
- **Answer questions:** Present questions to players within a predefined time limit, validating their answers and dynamically updating scores based on correctness and timeliness.
- **Track scores:** Provide real-time updates on player scores, allowing participants to monitor their progress and compare their performance with others in the game.
- **Display leaderboard:** Showcase the top players and their respective scores in a leaderboard format, fostering a spirit of healthy competition and encouraging players to strive for excellence.
- **Exit the game:** Implement functionality for players to exit the game gracefully, ensuring a smooth and intuitive user experience from start to finish.

One of the primary objectives is to design the user interface to be as accessible and straightforward as possible, enhancing the overall user experience with the multiplayer online quiz game.

Additionally, incorporate mechanisms for exception handling and signal management to address unexpected scenarios effectively. Ensure robust error handling for situations like network errors, invalid input, timeouts, game interruptions, player disconnections, and timeouts. Utilize signals to manage these events efficiently, maintaining the stability and reliability of the game environment throughout the gameplay session.

3. Dynamic Memory:

Utilize dynamic memory allocation for efficient resource management, including quiz questions, player scores, and other data structures. Ensure proper memory deallocation to prevent resource leaks.

4. Process Synchronization:

Implement semaphores to synchronize access to shared resources such as quiz data or player scores in your multiplayer online quiz game. Semaphores effectively coordinate multiple processes or threads, preventing race conditions and ensuring smooth game operation. By utilizing semaphores, you can control access to critical sections of code, allowing only one process or thread to access shared resources at a time. This ensures data integrity and prevents conflicts, contributing to a seamless and reliable gaming experience for all players.

5. Concurrency:

Integrate multi-threading into your multiplayer online quiz game to enable simultaneous player participation. By employing multi-threading, you can efficiently manage multiple players accessing shared resources concurrently while ensuring data integrity and synchronization. This approach optimizes system resource utilization, enhances responsiveness, and scalability, ultimately delivering a seamless and engaging gaming experience for all participants.

6. Networking:

Integrate socket-based networking capabilities to enable multiplayer functionality in your quiz game. Design a client-server architecture where the server manages game logic, while clients connect to participate. Utilize socket programming techniques to establish communication between the server and clients. This involves implementing protocols for data exchange, error handling mechanisms, and managing client-server interactions to ensure smooth gameplay. Additionally, handle socket-level errors, disconnections, and other network-related issues to maintain a stable and responsive gaming experience for all participants.

7. Data Storage and Management:

In your multiplayer online quiz game project, efficient data storage and management are essential for maintaining user credentials, quiz questions, and scores. To achieve this, you will utilize file-based storage mechanisms to mimic a database-like structure.

- User Credentials: A dedicated file will store user credentials for registration and login purposes. Each user's credentials, including username and hashed password, will be stored in this file. Proper encryption techniques will be employed to ensure the security of user data.
- Quiz Database: Another file or set of files will contain the quiz database, storing questions, answers, and other relevant information. To organize the quiz data efficiently, separate files may be used for each quiz category or subcategories. Each

quiz file will contain questions, possible answers, correct answers, and any other necessary details.

- **Scoreboard:** The scoreboard, displaying user scores for each quiz category, will be maintained in separate files. Like the quiz database, separate files may be utilized for each quiz category or subcategories to store score information. Scores will be updated and synchronized in real-time to reflect the latest results accurately.

By employing file-based storage mechanisms, you ensure data persistence and enable easy retrieval and manipulation of user credentials, quiz data, and scores. Proper organization and encryption techniques will be implemented to maintain data integrity, security, and efficiency throughout the operation of the quiz game.

General Requirements:

- Avoid system () functions for system stability.
- Use dynamic memory allocation for efficient resource management.
- Implement robust exception handling to manage errors and unexpected situations.
- Utilize process synchronization to ensure data consistency and prevent race conditions.
- The server must be capable of handling a minimum of five concurrent clients simultaneously.
- The server must be capable of accepting a command-line argument specifying the port number on which it will listen for incoming connections.
- The server should incorporate logging functionality to generate informative log messages, recording relevant events, interactions, and errors between the server and the client, including client connections, disconnections, quiz sessions, and any encountered errors during operation.
- Incorporate concurrency for simultaneous player participation.
- Implement networking for multiplayer functionality.
- Follow C code style guidelines for clean, readable, and maintainable code.
- Insert appropriate comments as needed in your code.
- Separate your code into multiple source and header files to keep them categorized and organized as possible. You should create a networking dynamic library that sends and receives commands.
- Utilize version control with Git for collaboration and change tracking.
- Include a comprehensive project report documenting design, functionality, testing procedures, and challenges.

Team Formation and Collaboration:

- The project consists of two to three students, organized into a total of five teams.
- Each team is required to collaborate with its members through GitHub, with team formations. The link will be facilitated via Moodle after the team members' confirmation.
- Team leaders will be designated within each team, and teams are encouraged to select a partner for collaboration.
- In cases where a team comprises only two members, the third student will be assigned to that team by me.
- NO TEAM with a single member is permitted under any circumstances, ensuring effective collaboration and distribution of workload among team members.

By including this information in the project document, you establish clear guidelines for team formation, collaboration, and leadership roles, promoting a structured and productive working environment for all participants involved in the project.

Deliverables:

- ✓ A Git repository containing all project source code. You should make frequent commit changes and pushes to GitHub. Include the README file that describes your project and how to run it.
- ✓ A well-documented project report describing the system's design, functionality, and testing procedures. The report should be organized as follows:
    - Cover page.
    - Table of contents
    - Introduction
    - Problem statement
    - Methodology and algorithms showing how your code works.
    - Sample tests and run to cover all the required tasks and functions of your system.
    - Conclusion which includes the lessons and problems the team has faced during project development.
    - Annex of code
    - References (if any)
- ✓ A comprehensive demonstration of the working project, showcasing key features and real-time capabilities. [Live Demo Presentation @ Wednesday 15/5/2024]

Project Submitting Phases:

Phase 1: Multiplayer Online Quiz Game Development (due 15/4/2024)

During this phase, teams will concentrate on establishing the fundamental functionalities and user interface of the multiplayer online quiz game. The primary focus will be on defining project requirements and objectives, as well as configuring the development environment (Linux/C, development tools). Key tasks include designing and implementing data structures to manage quizzes effectively, enabling the creation of new quiz sessions, updating quiz status in real-time, managing the quiz process, closing quizzes gracefully, and maintaining comprehensive quiz history. Additionally, teams will develop user-friendly views to facilitate browsing available quizzes, answering questions, selecting quiz categories, tracking scores, viewing leaderboards, and exiting the game smoothly.

Phase 2: Network and Concurrency Programming (due 14/5/2024)

In this phase, teams will shift their focus to network programming, concurrency, user authentication, and error handling for the multiplayer online quiz game. The primary objectives include introducing basic network programming concepts and protocols, establishing server-client communication channels, and implementing multi-threading techniques to support concurrent client interactions. Teams will also integrate network-based user authentication and access control mechanisms to ensure the security and integrity of the game environment. Additionally, efforts will be made to stress testing the system to ensure it can handle a minimum of five concurrently connected clients. Extensive testing will be conducted to validate network communication, concurrency management, and user authentication functionalities. Finally, teams will optimize the codebase, address any identified bugs, and ensure the overall smooth operation of the system.

Grade Distribution:

- *Functionality: 75%* (Evaluate the completeness and robustness of the system's features).
- *Code style and quality: 10%* (Assess the code's readability, organization, and adherence to coding standards).
- *Project report: 15%* (Review the clarity and completeness of the documentation, including explanations of design choices and testing procedures).

Note: team coordination, and adherence to best practices are essential to the success of this project.

Good luck!