Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

Студент: Муханов Андрей Ильич

Группа: М8О-209Б-23

Вариант: 12

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Структура коммуникации:
4. Реализованная функциональность
5. Положительные аспекты реализации
6. Вывод

**Репозиторий**

https://github.com/andreymuch/OC\_MAI

**Постановка задачи**

**Цель работы**

Целью работы является:

* Приобретение практических навыков в использовании знаний, полученных в течении курса
* Проведение исследования в выбранной предметной области

**Задание**

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Спроектировать консольную клиент-серверную игру на основе технологии MMF.

**Вариант**  
 12. «Быки и коровы» (угадывать необходимо числа). Общение между сервером и клиентом необходимо организовать при помощи memory map. При создании каждой игры необходимо указывать количество игроков, которые будут участвовать. То есть угадывать могут несколько игроков. Должна быть реализована функция поиска игры, то есть игрок пытается войти в игру не по имени, а просто просит сервер найти ему игру.

**Структура коммуникации:**

* + Клиент и сервер взаимодействуют через общую память (SharedMemory), содержащую:
    - Очередь запросов (requestQueue) для передачи команд от клиентов.
    - Массив ответов (clientResponses) для отправки результатов каждому клиенту.
  + Синхронизация доступа к ресурсам реализована через мьютексы (requestMutex, responseMutex).

**Реализованная функциональность**

1. Серверная часть:
   * Управление играми:
     + Создание игр с указанием максимального числа участников (create <game\_name> <max\_players>).
     + Подключение к конкретной игре (join <game\_name>) или автоматический поиск доступной игры (find).
     + Хранение активных игр в структуре std::map<std::string, Game>, где каждая игра содержит:
       - Секретное число (4 уникальные цифры).
       - Список игроков, их количество, лимит участников.
   * Механика игры:
     + Генерация секретного числа с уникальными цифрами (generateSecretNumber).
     + Подсчет быков и коров для каждого предположения (countBullsAndCows).
   * Обработка команд:
     + Выход из игры (exit).
     + Проверка корректности введенных чисел (длина = 4, только цифры).

Клиентская часть:

* + Подключение к серверу:
    - Регистрация клиента через уникальный clientId в массиве clientResponses.
    - Отображение присвоенного clientId для дальнейшего взаимодействия.
  + Отправка команд:
    - Поддержка команд: create, join, find, guess, exit.
    - Автоматическая отправка запросов в очередь requestQueue.
  + Получение ответов:
    - Чтение результатов из clientResponses[clientId] с синхронизацией через мьютекс.

**Положительные аспекты реализации**

1. Использование memory-mapped файлов:
   * Эффективная организация обмена данными между процессами через разделяемую память.
   * Минимизация накладных расходов по сравнению с сетевыми протоколами.
2. Синхронизация ресурсов:
   * Корректное применение мьютексов (requestMutex, responseMutex) для защиты:
     + Очереди запросов от одновременного доступа.
     + Ответов клиентов от конфликтов при записи/чтении.
3. Структуры данных:
   * Логичное разделение данных:
     + Request — для передачи команд.
     + ClientActive — для хранения состояния клиента.
     + SharedMemory — центральная точка обмена информацией.
4. Механика игры:
   * Алгоритм генерации секретного числа гарантирует уникальность цифр.
   * Корректный подсчет быков и коров (учет позиций и символов).
5. Многопользовательский режим:
   * Поддержка нескольких игроков в одной игре.
   * Возможность поиска доступных игр без указания имени (find).
6. Обработка ошибок:
   * Проверка на дубликаты игр при создании.
   * Валидация числа игроков (1–100).
   * Уведомления об ошибках (например, "Game is full", "Invalid command").

**Вывод**

Реализована базовая версия многопользовательской игры "Быки и коровы" с использованием memory-mapped файлов для межпроцессного взаимодействия. Код демонстрирует:

* Рабочую интеграцию клиента и сервера через разделяемую память.
* Корректную синхронизацию ресурсов.
* Поддержку создания игр, подключения, угадывания чисел и выхода.
* Механизмы валидации ввода и обработки ошибок.