Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовой проект по курсу
«Операционные системы»

Студент: Тарасов Егор Дмитриевич
Группа: М8О-209Б-23
Вариант: 1
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Содержание

Репозиторий

https://github.com/EgorX2000/os_labs/tree/main/course_project

Постановка задачи

Цель работы

- Приобретение навыков в использовании знаний, полученных в течение 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.
Вариант №8. Морской бой. Общение между сервером и клиентом необходимо организовать при помощи pipe'ов. Каждый игрок должен при запуске ввести свой логин. Для каждого игрока должна вестись статистика игр (сколько побед/поражений). Игрок может посмотреть свою статистику

Общие сведения о программе

Связь между клиентом и сервером поддерживается путём именованных pipe'ов. Все клиенты записывают команды в pipe сервера. Сервер же общается с клиентами через pipe посредством потоков.

Исходный код

Client.cpp

```
#include <iostream>
#include <windows.h>
#include <string>
#include "GameClient.h"

int main() {
    GameClient client;

    std::string username;
    std::cout << "Enter your username: ";
    std::cin >> username;

    client.sendRequest("LOGIN " + username);

    while (true) {
        std::cout << "Choose action: 1) Create Game 2) Join Game 3) View Stats: ";
```

```
        int choice;
        std::cin >> choice;

        if (choice == 1) {
            std::string gameName;
            std::cout << "Enter game name: ";
            std::cin >> gameName;
            std::string response = client.sendRequest("CREATE " + username + " " +
gameName);
            std::cout << response << std::endl;
        } else if (choice == 2) {
            std::string gameName;
            std::cout << "Enter game name: ";
            std::cin >> gameName;
            std::string response = client.sendRequest("JOIN " + username + " " +
gameName);
            std::cout << response << std::endl;
        } else if (choice == 3) {
            std::string response = client.sendRequest("STATS " + username);
            std::cout << response << std::endl;
        } else {
            std::cout << "Invalid choice!" << std::endl;
        }
    }

    return 0;
}
```

Server.cpp

```
#include <iostream>
#include <windows.h>
#include <string>
#include <thread>
#include <vector>
#include "GameManager.h"
#include "Player.h"

int main() {
    GameManager gameManager;

    while (true) {
        std::cout << "Waiting for clients..." << std::endl;

        HANDLE hPipe = CreateNamedPipeW(
            L"\\\\.\\pipe\\MyPipe",
            PIPE_ACCESS_DUPLEX,
            PIPE_TYPE_MESSAGE | PIPE_READMODE_MESSAGE | PIPE_WAIT,
            PIPE_UNLIMITED_INSTANCES,
            512, 512, 0, NULL);
```

```
        if (hPipe == INVALID_HANDLE_VALUE) {
            std::cerr << "Failed to create pipe. Error: " << GetLastError() <<
std::endl;
            continue;
        }

        if (ConnectNamedPipe(hPipe, NULL)) {
            std::thread([hPipe, &gameManager]() {
                char buffer[512];
                DWORD bytesRead;

                if (ReadFile(hPipe, buffer, sizeof(buffer), &bytesRead, NULL)) {
                    std::string message(buffer, bytesRead);
                    std::string response = gameManager.processRequest(message);

                    DWORD bytesWritten;
                    WriteFile(hPipe, response.c_str(), response.size(),
&bytesWritten, NULL);
                }

                CloseHandle(hPipe);
            }).detach();
        }
    }

    return 0;
}
```

Player.h

```
#ifndef PLAYER_H
#define PLAYER_H

#include <string>

class Player {
public:
    Player() : username(""), wins(0), losses(0) {} // Конструктор по умолчанию
    Player(const std::string& username);
    int getWins() const;
    int getLosses() const;
    void addWin();
    void addLoss();

private:
    std::string username;
    int wins;
    int losses;
};

#endif
```

Player.cpp

```
#include "Player.h"

Player::Player(const std::string& username) : username(username), wins(0),
losses(0) {}

int Player::getWins() const {
    return wins;
}

int Player::getLosses() const {
    return losses;
}

void Player::addWin() {
    wins++;
}

void Player::addLoss() {
    losses++;
}
```

GameManager.h

```
#ifndef GAMEMANAGER_H
#define GAMEMANAGER_H

#include <string>
#include <map>
#include "Player.h"

class GameManager {
public:
    std::string processRequest(const std::string& request);

private:
    std::map<std::string, Player> players;
    std::map<std::string, std::string> games;

    std::string handleLogin(const std::string& username);
    std::string handleCreateGame(const std::string& username, const std::string&
gameName);
    std::string handleJoinGame(const std::string& username, const std::string&
gameName);
    std::string handleStats(const std::string& username);
};

#endif
```

GameManager.cpp

```cpp
#include "GameManager.h"

std::string GameManager::processRequest(const std::string& request) {
    // Пример обработки запроса
    if (request.find("LOGIN") == 0) {
        std::string username = request.substr(6);
        return handleLogin(username);
    } else if (request.find("CREATE") == 0) {
        size_t pos = request.find(' ', 7);
        std::string username = request.substr(7, pos - 7);
        std::string gameName = request.substr(pos + 1);
        return handleCreateGame(username, gameName);
    } else if (request.find("JOIN") == 0) {
        size_t pos = request.find(' ', 5);
        std::string username = request.substr(5, pos - 5);
        std::string gameName = request.substr(pos + 1);
        return handleJoinGame(username, gameName);
    } else if (request.find("STATS") == 0) {
        std::string username = request.substr(6);
        return handleStats(username);
    } else {
        return "INVALID_REQUEST";
    }
}

std::string GameManager::handleLogin(const std::string& username) {
    players[username] = Player(username);
    return "LOGIN_SUCCESS";
}

std::string GameManager::handleCreateGame(const std::string& username, const
std::string& gameName) {
    games[gameName] = username;
    return "GAME_CREATED:" + gameName;
}

std::string GameManager::handleJoinGame(const std::string& username, const
std::string& gameName) {
    if (games.find(gameName) != games.end()) {
        return "JOIN_SUCCESS:" + games[gameName];
    } else {
        return "GAME_NOT_FOUND";
    }
}

std::string GameManager::handleStats(const std::string& username) {
    if (players.find(username) != players.end()) {
        return "STATS:" + std::to_string(players[username].getWins()) + ":" +
std::to_string(players[username].getLosses());
    } else {
        return "USER_NOT_FOUND";
```

```
    }
}
```

GameClient.h

```
#ifndef GAMECLIENT_H
#define GAMECLIENT_H

#include <string>
#include <windows.h>

class GameClient {
public:
    std::string sendRequest(const std::string& request);

private:
    HANDLE connectToServer();
};

#endif
```

GameClient.cpp

```
#include "GameClient.h"
#include <windows.h>
#include <iostream>

std::string GameClient::sendRequest(const std::string& request) {
    HANDLE hPipe = connectToServer();
    if (hPipe == INVALID_HANDLE_VALUE) {
        return "Failed to connect to server.";
    }

    DWORD bytesWritten;
    if (!WriteFile(hPipe, request.c_str(), request.size(), &bytesWritten, NULL)) {
        return "Failed to send request.";
    }

    char buffer[512];
    DWORD bytesRead;
    if (!ReadFile(hPipe, buffer, sizeof(buffer), &bytesRead, NULL)) {
        return "Failed to read response.";
    }

    CloseHandle(hPipe);
    return std::string(buffer, bytesRead);
}

HANDLE GameClient::connectToServer() {
    return CreateFileW(
        L"\\\\.\\pipe\\MyPipe",
        GENERIC_READ | GENERIC_WRITE,
```

```
        0, NULL, OPEN_EXISTING, 0, NULL);
}
```

Демонстрация работы программы

// server
PS C:\Users\Xiaomi\Desktop\kpp\build> ./server

Waiting for clients...

Waiting for clients...

Waiting for clients...

Waiting for clients...

Waiting for clients...

Game Game1 started between Player1 and Player2

//client1
PS C:\Users\Xiaomi\Desktop\kpp\build> ./client
Enter your username: Player1
Choose action: 1) Create Game 2) Join Game 3) View Stats: 1
Enter game name: Game1
GAME_CREATED:Game1
Choose action: 1) Create Game 2) Join Game 3) View Stats: 3
STATS:0:0

//client2
PS C:\Users\Xiaomi\Desktop\kpp\build> ./client
Enter your username: Player2
Choose action: 1) Create Game 2) Join Game 3) View Stats: 2
Enter game name: Game1
JOIN_SUCCESS:Player1

Выводы

Именованные pipe отлично справляются со своей задачей коммуникации между процессами в силу своей простоты и удобства использования. Был получен опыт разработки консольной клиент-серверной игры. Благодаря этому я понял, как происходит процесс общения между клиентом и сервером.