Tadeusz Springer 160116

Sprawozdanie sk2

Temat

1. Opis projektu

Celem projektu było stworzenie sieciowej, turowej gry logicznej (szachy), działającej w architekturze klient-serwer. System umożliwia prowadzenie wielu równoległych rozgrywek pomiędzy parami graczy. Aplikacja pozbawiona jest graficznego interfejsu użytkownika – cała interakcja odbywa się w konsoli za pomocą tekstu i ASCII reprezentującego planszę.

Kluczowym założeniem projektu jest to, że klient odpowiada jedynie za wyświetlanie otrzymanych danych i pobieranie ruchu od użytkownika, natomiast cała logika gry, walidacja ruchów, detekcja szacha/mata oraz zarządzanie stanem planszy odbywa się po stronie serwera.

Użyta technologia:

- Język programowania: C++ (z wykorzystaniem bibliotek standardowych C dla obsługi sieci).
- API Sieciowe: BSD Sockets (<sys/socket.h>, <netinet/in.h>) – niskopoziomowa obsługa gniazd w systemie Linux/Unix.
- Wielowątkowość: Biblioteka POSIX Threads (<pthread.h>) do obsługi wielu gier jednocześnie.
- Protokół: TCP/IP (gwarancja dostarczenia pakietów i kolejności danych).
- Struktury danych: Klasy i tablice dwuwymiarowe do reprezentacji planszy oraz wektory/listy do zarządzania stanem gry.

2. Opis komunikacji pomiędzy serwerem i klientem

Komunikacja odbywa się za pomocą protokołu tekstowego przesyłanego przez gniazda strumieniowe (SOCK_STREAM).

Schemat nawiązywania połączenia:

1. Serwer nasłuchuje na porcie 1100.
2. **Klient A** łączy się (connect). Serwer umieszcza go w poczekalni i wysyła komunikat "Czekanie na drugiego gracza...".
3. **Klient B** łączy się. Serwer paruje gniazda Klienta A i Klienta B w strukturę GameSession.
4. Serwer uruchamia nowy wątek (pthread_create), przekazując mu sesję gry. Główny wątek wraca do nasłuchiwania.

Pętla gry (Protokół): Wymiana danych w trakcie rozgrywki przebiega następująco:

1. **Serwer (Wysyłanie stanu):** Generuje tekstową reprezentację planszy (współrzędne a-h, 0-7) oraz instrukcję.
    - Do gracza aktywnego wysyła: [PLANSZA] + "Twój ruch >"
    - Do gracza oczekującego wysyła: [PLANSZA] + "Czekaj na ruch przeciwnika..."
2. **Klient (Akcja):**
    - Odbiera bufor i wyświetla go na ekranie.
    - Jeśli otrzymał prośbę o ruch, pobiera od użytkownika ciąg znaków (np. e 6 e 4) i wysyła go do serwera.
3. **Serwer (Walidacja):**
    - Parsuje wiadomość.
    - Sprawdza poprawność ruchu w klasie ChessGame.
    - Jeśli ruch jest poprawny: aktualizuje planszę, sprawdza warunek wygranej (MAT), przełącza turę i wraca do kroku 1.
    - Jeśli ruch jest błędny: wysyła komunikat błędu tylko do aktywnego gracza i prosi o ponowny ruch (bez odświeżania ekranu przeciwnika).

Zakończenie: W przypadku wykrycia mata lub rozłączenia się jednego z graczy, serwer wysyła komunikat końcowy (np. "SZACH MAT", "Walkower") i zamyka gniazda.

3. Podsumowanie

Projekt został zrealizowany zgodnie ze standardem BSD Sockets. Kluczowym elementem implementacji jest logika gry w klasie ChessGame. Każdy wątek gameThread posiada własny obiekt ChessGame, dzięki czemu gracze z różnych par nie ingerują w swoje rozgrywki.

Walidacja ruchu działa na zasadzie wykonania każdego możliwego ruchu zgodnego z ruchami pionków i wykonuje ruch na próbę, sprawdza, czy król nie został odsłonięty na atak (funkcja isKingInCheck), i jeśli tak, cofa zmiany na planszy, uznając ruch za nielegalny. Jest to rozwiązanie bezpieczniejsze i prostsze niż przewidywanie wszystkich możliwych konsekwencji przed wykonaniem ruchu.

Co sprawiło trudność:

1. **Synchronizacja widoków:** Największym wyzwaniem było obsłużenie sytuacji błędnych ruchów. Początkowo, gdy gracz A wykonał błąd, plansza odświeżała się również u gracza B, co powodowało niepotrzebne miganie konsoli. Rozwiązano to poprzez dodanie flagi updateOpponent w serwerze, która blokuje wysyłanie danych do przeciwnika, jeśli ruch nie został zatwierdzony.
2. **Obsługa wejścia/wyjścia:** Standardowa funkcja scanf powodowała problemy przy wczytywaniu poleceń ze spacjami (np. "e 6 e 4"). Konieczne było przejście na funkcję fgets po stronie klienta oraz precyzyjne parsowanie sscanf po stronie serwera.
3. **Konwersja współrzędnych:** Logika gry operuje na indeksach tablicy [y][x] (0-7), natomiast interfejs użytkownika na notacji szachowej (a-h, 0-7). Konieczne było zaimplementowanie warstwy tłumaczącej litery na indeksy oraz obsługa błędów formatowania (np. wpisanie znaku spoza zakresu).