Programowanie Sieciowe

Projekt:

“Gra sieciowa - Statki”

Język programowania: C

Zespół:
Karolina Rapacz
Wojciech Gałecki
Michał Szczepaniak-Krupowski

Prowadzący:
dr. inż. Janusz Gozdecki **14.02.2020**

1. Funkcjonalność programu

Program pozwala na grę w statki jednej lub wielu parom klientów. Wystarczy, że dany gracz będzie posiadał na swoim komputerze odpowiedni program, tj. client_battleship_game.c. Użytkownik łączy się z serwerem poprzez wpisanie w konsolę "./client_battleship_game <adres IP serwera>". Serwer pracuje jako demon z logowaniem do plików systemowych, przekazując koordynaty pól do zbicia oraz inne komunikaty (np. o odłączeniu się jednego z klientów). W konsoli klienta wyświetlają się komunikaty odnośnie stworzenia gniazda i nawiązaniu połączenia oraz te dotyczące samej gry, do których użytkownik musi się stosować. Najpierw użytkownik, który podłączył się wcześniej, czeka na użytkownika "do pary", aby rozpocząć grę. Następnie ustawiają statki na swojej planszy. Klient który podłączył się do serwera pierwszy wykonuje pierwszy ruch. Każdy z nich wykonuje ruch, jeśli był on trafiony, użytkownikowi przysługują kolejne, jeśli nie, strzał wykonuje przeciwnik. Gra kończy się w momencie, gdy jeden z graczy zniszczy wszystkie statki przeciwnika (tzn. wszystkie pola ze statkami zostały trafione). Przez serwer zostają rozesłane odpowiednie komunikaty o tym informujące, a także wiadomość o zakończeniu gry i rozłączeniu przeciwników.

2. Budowa programu

Program powstał w oparciu o protokół TCP, celem zabezpieczenia przed utratą przesyłanych pakietów. Klient jest "zwykłym" programem, wyświetlającym dane w konsoli, pobierającym input od użytkownika oraz komunikującym się z serwerem. Korzysta z napisanych przez nas funkcji:

- ***translate_position(), translate_position_game_core()*** - translacja koordynat podanych przez użytkownika, jako wielkie litery alfabetu (np. 'A') na zmienne typu int, celem umieszczenia pod odpowiednim indeksem strzału, statku itd. w tablicy 2D.
- ***error_handler()*** - obsługa błędów spowodowanych wpisaniem przez użytkownika niepoprawnych koordynat.
- ***ship_initialize()*** - funkcja obsługująca wypełnianie przez klientów swoich plansz statkami, kontrolę poprawności oraz wykonanie zapisu. Gracze do dyspozycji mają w sumie 10 okrętów o różnych długościach.
- ***game_core()*** - funkcja odpowiadająca za przeprowadzenie rozgrywki pomiędzy graczami, czyli odbiór inputu od użytkownika z koordynatami, obsługę błędów, określenie czy statek został trafiony oraz komunikację z serwerem, celem odebrania/wysłania pozycji do strzału, zakończenia/przerwania gry.

W funkcji main() tworzymy tablice dla graczy, gniazdo oraz bindujemy je z portem. Następnie wykonywane są odpowiednie funkcje, celem przeprowadzenia rozgrywki. Na końcu deskryptor gniazda jest zamykany.

Serwer został stworzony jako wielowątkowy proces demona (demona inicjalizuje funkcja ***daemon_init()***). Jest tylko pośrednikiem w wymianie informacji pomiędzy klientami. Operuje na mechanizmie epoll() level-triggered. Przydziela każdemu z klientów osobny deskryptor i naprzemiennie odczytuje informacje przesłane od klienta. Posiada on włączoną flagę SO_REUSEADDR oraz wyłączoną IPV6_V6ONLY.

Komunikacja odbywa się głównie poprzez przesłanie 40-elementowej tablicy intów, w której umieszczone są kolejne koordynaty statków do zestrzelenia lub inne wiadomości, tj. ***win_message*** - o wygraniu gry przez przeciwnika, czy liczba 60 informująca o zerwaniu połączenia przez drugiego gracza.

Zaimplementowany został również system obsługi błędów związanych z niepoprawnymi danymi wprowadzanymi przez użytkowników - od użytkownika oczekuje się wpisania danych w dokładnie takim formacie, jak to zostało określone w wyświetlonej instrukcji (np. przy stawianiu dwumasztowca, jeśli chcemy go umieścić na polach A2 i A3 należy wpisać “A2-A3” lub “A3-A2”, koniecznie wielką literą oraz z użyciem myślnika bez spacji). W przeciwnym wypadku w konsoli wyświetli się komunikat o niepoprawności danych, a użytkownik zostanie poproszony o ponowne ich wprowadzenie. Nie ma limitu pomyłek, czynność może być powtarzana do momentu poprawnego wprowadzenia.

3. Napotkane problemy

Podczas tworzenia programu spotkaliśmy się z anomaliami w jego funkcjonowaniu, jedną z nich zgłosiliśmy prowadzącemu, który przyznał, że nie ma ona jasno określonej przyczyny (samoistna zmiana wartości wpisanej do zmiennej).

Początkowo gra była tworzona w oparciu o protokół UDP, jednak okazał się on zbyt zawodny i obciążający system, co wymusiło na nas przerobienie programu na protokół TCP.

4. Podział zadań

W. Gałecki: Opracowanie serwera i poprawki w logice gry
K. Rapacz: Opracowanie logiki działania gry
M.Szczepaniak: Testowanie programu na każdym etapie jego tworzenia, optymalizacja kodu i poprawa jego czytelności