Gra w Statki

Dokumentacja

Franciszek Zdobylak

23 stycznia 2019

1 Komunikacja między procesami

Programy komunikują się przy użyciu plików kolejkowych. Do ich implementacji użyłem pliku pokazywanego podczas wykładu Wstępu do C (`lin-fifo.c`).

Do obsługi użyłem swojego modułu, który wysyła specjalnie przygotowane wiadomości do bliźniaczego procesu używając interfejsu wytworzonego w poprzednio wspomnianym pliku.

Wszystkie wiadomości są postaci mx_1x_2s, gdzie
m – tryb (s - zapytanie, f - odpowiedź, n - nowa gra, g - poddanie się),
x_1, x_2 – współrzędnie strzału (jeśli wiadomość nie wymaga tego to dowolne znaki),
s – status pola oznaczonego przez współrzędne x_1, x_2. Dostępne funkcje:

- `sendMove(PipesPtr, Shoot)` - wysyła zapytanie o dane pole na planszy
- `sendFeedback(PipesPtr, Shoot, Status)` - wysyła odpowiedź na zapytanie poprzedniej funkcji
- `sendSignal(PipesPtr, int)` - wysyłanie sygnału nowej gry(1), poddania się (0)
- `sendReveal(PipesPtr, Shoot, Status)` - wysyłanie inforamcji o polu na planszy po zakończeniu gry (odkrywanie planszy)
- `getMessage(PipesPtr, char*)` - odbieranie wiadomości i zapisywanie w stringu podanym jako parametr

2 Logika gry

Do obsługi gry służy moduł znajdujący się w plikach `game.c i game.h`. Są w nim opisane wszystkie funkcje oraz typy danych potrzebne do toczenia rozgrywki.

2.1 Typy danych

- Typ wyliczeniowy `Status` - służy do zaznaczania statnów na planszy. Składa się z elementów: NOT_SHOOT, SHIP, MISSED, HIT, SUNK, MY_HIT, UNKNOWN.
- Struktura `Shoot` - para liczb, służy do określenie pozycji na planszy
- Struktura `Ships` - przechowuje ilość statków poszczególnej długości, długość najdłuższego oraz liczbę wszystkich statków i pozostałych (niezestrzelonych)
- Typ `Board` - tablica 2-wymiarowa 10x10, przechowuje opis planszy (zgodnie z wartościami typu `Status`)

2.2 Ważniejsze funkcje

- `bool isSunk(Shoot s, Board b)` - zwraca prawdę jeśli statek, którego część znajduje się na pozycji podanej jako pierwszy argument, jest zatopiony (tzn. każde pole na którym się znajduje oznaczone jest jako trafione
- `int markSunk(Shoot s, Board b)` - zaznacza cały statek jako zatopiony (zmienia status z HIT/MY_HIT na SUNK), zwraca długość zaznaczonego statku
- `bool placeShip(int length, int orientation, Shoot, Board)` - próbuje ustawić statek o długości `length` i orientacji (1 - poziomej, 0 - pionowej) zaczynając na polu s na planszy b. Jeśli się da to zwraca prawdę i zaznacza statek na planszy. W przeciwnym wypadku zwraca fałsz.
- `void removeShip(Shoot pos, Board b)` - usuwa statek którego część znajduje się na pozycji `pos` z planszy b.

3 Tworzenie planszy

Tworzenie plansz oraz ładowanie ilości statków z pliku umieszczone jest w module w plikach `boards.c i boards.h`. Dostępne funkcje:

- `getShips(Ships *s)` - pobiera ilość statków z pliku `ships.len`. Przy wczytywaniu pomija linie oznaczone znakiem # (są to komentarze).
- `randBoard(Board b, Ships *s, char name)` - funkcja wywoływana przez resztę modułów. Generuje planszę oraz wczytuje długości statków z pliku.
- `genBoard(Board b, Ships s, char name)` - funkcja generująca plnnaszę z ilością statków, które zostały podane jako jeden z parametrów. Znak `name` służy do lepszego generowania losowych liczb. Algorytm wstawiania statków jest prosty. Losujemy miejsce w którym statek ma się zaczynać oraz jego orientację. Jeśli statek się w takim miejscu nie zmieści to sprawdzamy czy zmieści się w innej orientacji w tym miejscu. Jeśli znów nie, to przesuwamy się o jedno miejsce dalej. Jeśli statek nie może zostać położony w żadny miejscu, to generujemy planszę od nowa. Statków nie jest za dużo, więc praktycznie jest to niemożliwe aby algorytm się zapętlił w nieskończoność.

4 Tworzenie okna

Okno tworzę przy pomocy biblioteki GTK 3.0. Całe tworzenie okna głównego oraz okna tworzenia planszy jest zaimplementowane w module `window.c`.

4.1 Ważniejsze Funkcje

- `show_alert(char *alert)` - funkcja do wyświetlania okna dialogowego z komunikatem `alert`
- `get_move()` - funkcje do obsługi klikania guzików na planszy
- `create_board()` - wyświetla okienko wyboru planszy
- `create_ship()` - próbuje ustawić statek na polach zaczynających się od kliknięgo (ta funkcja jest podłączona do przycisków na planszy ustawiania statków)
- `refresh()` - funkcja do pobierania wiadomości z pliku kolejkowego w odpowiednich odstępach czasowych. W zależności od otrzymanej wiadomości wywołuje odpowiednie czynności (rozpoczęcie nowej gry, sprawdzenie pola na planszy itp.)
- `chage_button(GtkWidget *button, Status stat)` - zmienia grafikę na przycisku `button` w zależności od statusu `stat`
- `epilog(char option)` - zakończenie gry w zależności od opcji. (`'w'` - wyrana, `'L'` - przegrana, `'g'` - przeciwnik poddał się, `'l'` - poddanie się)
- `update_board(char option)` - aktualizuje planszę w zależności od opcji (`'m'` - moją, `'o'` - przeciwnika, `'c'` - ustawiania statków)
- `main()` - tworzy główne okno

4.2 Struktury i zmienne

- `struct statistics_struct` - przechowuje wskaźniki na obiekty typu GTK_LABEL służące do wyświetlania statystyk
- `struct length_panel` - przechowuje przyciski zmieniające długość stawianego statku w oknie tworzenia planszy
- `struct orientation_panel` - przechowuje przyciski zmieniające orientację stawianego statku w oknie tworzenia planszy
- `bool my_round` - `true` jeśli trwa runda gracza
- `bool game_run` - `true` jeśli gra jest rozpoczęta
- `bool end_of_game` - `true` jeśli gra się skończyła