Koncepcja projektu nr 3

„Synchronizacja - semafory”

1. Ogólne rozwiązanie problemu
   1. Zaimplementowanie procesu generującego samochody. Co jakiś czas (definiowany przez użytkownika) będzie generowany samochód (tzn. losowana będzie droga wjazdowa, zjazdowa i priorytet, a także dołączane będzie id) i wrzucana jest do jednej z trzech kolejek wjazdowych. Całkowita ilość samochodów w symulacji jest zapisana jako stała w kodzie
   2. Implementacja skrzyżowania jako konsumenta i zarazem producenta. Pojemność skrzyżowania wynosi 1 samochód. Skrzyżowanie jako konsument pobiera samochód z drogi wjazdowej bądź też z drogi między samochodami (pierwszeństwo mają zawsze samochody priorytetowe a następnie te z drogi wjazdowej, jeśli będą oczekiwać dwie karetki pierwszeństwo ma ta z drogi wjazdowej), ale nie może pobrać elementu z pustych kolejek (mechanizm konsumenta). Samochód na skrzyżowaniu spędza około 1 sekundy. Następnie sprawdzane jest czy zjazd elementu jest w danym skrzyżowaniu, jeśli tak element bezpośrednio zjeżdża ze skrzyżowania, jeśli nie element zostaje przekazany do drogi między skrzyżowaniami, ale nie zostaje przekazany jeśli droga ta ma aktualnie 10 pojazdów (mechanizm producenta).
   3. Implementacja kolejki (drogi między skrzyżowaniami) aby miała możliwość ustawienia maksymalnego rozmiaru, łatwą metodę sprawdzenia czy jest pusta, obsługę samochodów priorytetowych (karetek), polegającą na wstawianiu normalnych samochodów na koniec kolejki a karetek na początek.
   4. Drogi wjazdowe i między skrzyżowaniami są zasobami współdzielonymi
   5. Semafory zostaną użyte w skrzyżowaniach do realizacji mechanizmu producent-konsument aby uniknąć wyścigów.
   6. W momencie zakończenia symulacji (działania main) wysyłany jest sygnał kill do procesów crossA, crossB i crossC, a następnie zwalniana jest pamięć współdzielona.
2. Lista tworzonych plików:
   1. queue.h – plik nagłówkowy zawierający interfejs kolejki
   2. proQueue.h – plik nagłówkowy zawierający interfejs kolejki z generowanymi samochodami
   3. car.h – plik zawierający definicję pojazdu
   4. main.c – główne ciało programu, odpowiadające za prawidłową inicjalizację zasobów współdzielonych i semaforów, a także generację samochodów
   5. cross.h – plik nagłówkowy zawierający interfejs skrzyżowania
   6. crossA.c – plik z realizacją skrzyżowania A
   7. crossB.c – plik z realizacją skrzyżowania B
   8. crossC.c – plik z realizacją skrzyżowania C
3. Wizualizacja.
   1. Uruchamiamy główny program i skrzyżowania w oddzielnych konsolach, ustawiamy w głównym programie ilość generowanych samochodów i czas z jakimi będą się pojawiać.
   2. W każdym ze skrzyżowań jeśli samochód wjeżdża lub zjeżdża z niego, wyświetlany jest odpowiedni komunikat informujący jaki samochód wjechał na skrzyżowanie i do której drogi zjechał.

A C

B