Systemy Operacyjne – laboratorium

Ćwiczenie 3 – Semafory

1. Treść zadania

Celem ćwiczenia jest napisanie programu w języku C++ w środowisku systemu Linux realizującego organizację bufora komunikacyjnego przy pomocy monitora. Bufor komunikacyjny jest strukturą danych mieszczącą maksymalnie M elementów jednakowego typu. W zaproponowanej implementacji należy zapewnić synchronizację:

- nie dopuścić do czytania z pustego bufora;
- nie dopuścić do zapisu do pełnego bufora;
- zadbać o „nie przeszkadzanie” sobie procesów korzystających z bufora.

2. Przyjęte założenia

- przykładowa (modyfikowalna) długość kolejki – 100
- typy procesów korzystających z bufora:
  - producent (dodaje wartości do bufora)
  - konsument (pobiera wartości z bufora)
- typ przechowywanych danych – dowolny
- kolejność odczytu i zapisu danych – kolejka LIFO

3. Proponowane rozwiązanie

W celu rozwiązania zadania zamierzam zaimplementować klasę szablonową **Bufor**. Klasa będzie parametryzowana typem przechowywanych wartości. Aby zapewnić odpowiednią synchronizację (opisaną w punkcie pierwszym), klasa **Bufor** będzie dziedziczyć po klasie **Monitor** znajdującej się w dołączonym do treści zadania pliku **monitor.h**. Klasa **Bufor** będzie zawierać nastepujące metody:

- **konstruktor** – odpowiada za odpowiednią inicjalizację wykorzystywanych zmiennych oraz alokację odpowiedniej ilości pamięci na bufor;
- **destruktor** – odpowiada za dealokację pamięci przeznaczonej na bufor;
- **push** – metoda dodająca jeden element do bufora;
- **pop** – metoda usuwająca jeden element z bufora.

4. Testowanie

W celu implementacji i przetestowania omawianego bufora, stworzę plik nagłówkowy z definicją klasy **Bufor**, a następnie plik wykonywalny odpowiedzialny za utworzenie obiektu tej klasy oraz wywołanie określonej ilości procesów symbolizujących czytelników oraz pisarzy. Wspomniane wywoływanie będzie kontrolowane interaktywnie, tzn. użytkownik będzie w stanie w każdym momencie zdecydować jakiego typu procesy, jakie ich ilości oraz w jakiej kolejności uruchomić. Dzięki temu istnieje możliwość przetestowania wszystkich sytuacji.

Autor: Maciej Suchecki – grupa 3I4