Koncepcja wstępna do laboratorium nr 4
z przedmiotu KSO
Synchronizacja procesów z wykorzystaniem monitorów

Agnieszka Hermaniuk

Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych

17 grudnia 2021

Spis treści

1 Treść zadania

Należy zrealizować typ "bufor komunikacyjny". W czasie implementacji należy zapewnić synchronizację:

- nie dopuścić do czytania z pustego bufora
- nie dopuścić do zapisu do pełnego bufora
- zadbać o "nie przeszkadzanie sobie" procesów zapisujących do bufora i procesów czytających z bufora

Dodatkowe kryteria:

- liczba buforów: 3
- każdy bufor ma inną liczbę elementów aktualnie przechowywanych: 1-najwięcej, 2-mniej, 3-najmniej
- każdy bufor ma inną kolejność zapisu do i odczytu z bufora (ustaloną przez studenta)

Projekt ma być zrealizowany w języku C++ na środowisku Linux.

2 Koncepcja teoretyczna

W realizacji zadania należy stworzyć 3 bufory w postaci kolejek. Przyjęte wartości charakteryzujące te bufory przedstawiam w tabeli poniżej.

	Max liczba elementów przechowywanych	Kolejność zapisu i odczytu	Liczba elementów jednocześnie dostępnych*
bufor 1	10	FIFO	3
bufor 2	8	LIFO	2
bufor 3	6	FIFO	1

*Tak jak w rozwiązaniu laboratorium nr 3, nie będzie to "dosłownie" dostęp jednoczesny, a po prostu liczba elementów dokładanych/zdejmowanych z bufora przez proces w pętli.

Zaimplementowany zostanie problem producenta i konsumenta, mających dostęp do wszystkich trzech buforów. Producent będzie produkował - dodawał do buforów nowe elementy, a konsument konsumował - usuwał elementy z bufora. Producent może zapisywać do bufora 1, 2 lub 3 elementy, zależnie od tego, do którego bufora zapisuje i czy jest w nim odpowiednia ilość miejsca. Konsument może zdejmować odpowiednio taką samą liczbę elementów.

Pracę systemu należy prawidłowo zorganizować:

1. Producent musi czekać, gdy bufor jest pełny. Gdy są wolne miejsca, zostaje odblokowany i może umieszczać w buforze elementy.
2. Gdy bufor jest pusty, konsument musi czekać. Gdy pojawią się elementy, zostaje odblokowany i może pobierać z bufora elementy.
3. Dwa procesy nie mogą mieć dostępu w tym samym czasie do bufora.

Kontrolę dostępu do zasobów zapewnią monitory. Monitor zawierać będzie deklaracje zmiennych:

- int count - liczba zajętych slotów w buforze
- condition full
- condition empty

a także ciała funkcji, które będą wykonywać operacje na tych zmiennych:

- enter
- remove

Sekcja krytyczna zostaje więc przeniesiona do monitora. Procesy nie mają bezpośredniego dostępu do zmiennych (w przeciwieństwie do implementacji z semaforami), a jedynie mogą wywoływać funkcje, które będą dokonywały zmian na zmiennych.

Zmienne typu condition mogą być użyte tylko w operacjach wait() oraz signal(), które blokują proces w monitorze, jeśli z jakiegoś powodu nie może on wykonać zadania (w przypadku producenta, gdy bufor jest pełny, a w przypadku konsumenta - gdy jest pusty). Zablokowany proces wykonuje operację wait na zmiennej warunkowej, wtedy inny proces może wejść do sekcji krytycznej, a na wyjściu z niej wywołać operację signal na zmiennej warunkowej, w celu obudzenia zawieszonego procesu.

Poniżej przedstawiam uproszczoną implementację rozwiązania problemu dla jednego bufora:

```
monitor Buffer
    condition full, empty;
    int count;

  procedure enter();
  {
    if (count == N) wait(full);
    put_item(widget);
    count = count + 1;
    if (count == 1)  signal(empty);
  }

  procedure remove();
  {
    if (count == 0)  wait(empty);
    remove_item(widget);
    count = count - 1;
    if (count == N-1)  signal(full);
  }

  count = 0;
  end monitor;

  Producer();
  {
    while (TRUE)
   {
      make_item(widget);
      Buffer.enter;
   }
  }

  Consumer();
  {
    while (TRUE)
    {
      Buffer.remove;
      consume_item;
   }
  }
```

Do napisania programu najprawdopodobniej skorzystam z pliku monitor.h znajdującego się na stronie prowadzącego.

3 Testowanie

W ramach testowania uruchomione zostaną procesy producenta oraz konsumenta, wykonujące operacje zdejmij/włóż w pętli. Wypisywane będą statusy z dokonanych operacji wraz ze sprawdzeniem ewentualnych błędów (np. komunikat, że konsument dotarł do sekcji krytycznej pomimo pustego bufora). Dla rozróżnienia buforów, każdy z nich będzie przechowywał inne wartości:

- bufor 1 - cyfry
- bufor 2 - małe litery
- bufor 3 - duże litery