Wojciech Sitek

Koncepcja SOI – zadanie 3

# Treść zadania

* Napisać program w języku C wykorzystujący działanie semaforów do symulacji bufora z producentami i konsumentami
* Dany jest jeden bufor 9-elementowy FIFO
* Producent A produkuje jedną literę
* Producent B produkuje trzy litery
* Konsument A czyta jedną literę
* Konsument B czyta dwie litery
* Element jest usuwany z bufora po odczytaniu przez konsumenta
* Liczba elementów w buforze po przeczytaniu przez dowolnego konsumenta nie może spaść poniżej 3.

# Koncepcja rozwiązania zadania

* Implementacja struktury person, przechowującej:
  + Tablicę flag *cantPass* dla każdego semafora (gdy inna osoba podnosi semafor, pozwala wejść do strefy krytycznej tej osobie przez podniesienie flagi cantPass dla tego semafora)
  + Wartość całkowitą *jump*, określającą, o ile zmniejszamy/zwiększamy semafor (gdy np. osoba produkuje trzy litery, jump=3). Jest uaktywnione tylko wtedy, gdy semafor.jump=0
  + Nazwę osoby, potrzebną do wyświetlenia informacji o niej w programie.
* Implementacja makra struktury Queue dla typu char (bufor 9-elementowy FIFO) lub person \* (kolejka oczekująca semafora) z funkcjami obsługującymi strukturę:
  + add\_queue(Queue\*, TYPE element) – dodanie elementu na koniec kolejki,
  + front\_queue(Queue \*) – zwrócenie pierwszego elementu bez jego usunięcia,
  + pick\_queue(Queue \*) – zwrócenie pierwszego elementu i jego usunięcie z kolejki
* Implementacja struktury *semaphore*, przechowującej:
  + Wartość semafora
  + Wartość całkowitą *jump*, o ile skacze wartość semafora przy jego podniesieniu/opuszczeniu
  + ID semafora, niezbędne przy określaniu zmiennej cantPass struktury person
  + Kolejkę osób oczekujących na podniesienie semafora
  + Wartość całkowitą *raise*, określającą, o ile już podniósł się semafor, w przypadku gdy na wejście do strefy krytycznej oczekuje osoba o jump>raise. Wtedy semafor jest podnoszony dopiero wtedy, gdy jump osiągnie wartość raise
* Zdefiniowanie semaforów:
  + Mutex – sekcja krytyczna bufora
  + Empty – ilość pustych miejsc w buforze
  + Full – ilość pełnych miejsc w buforze
  + Finish – czy kończymy program
  + userMutex – sekcja krytyczna komunikacji wątków z użytkownikiem
* Implementacja funkcji down(semaphore \*s, person \*p), up(semaphore \*s, person \*p):
  + Funkcja down opuszcza wartość semafora o jump. Następnie gdy po opuszczeniu semafora jego wartość jest mniejsza od 0 (inna osoba jest w strefie krytycznej), osoba zostaje dodana do kolejki oczekującej i wchodzi w pętlę oczekującą na zmianę jej pola *cantPass* przez funkcję *up* innej osoby.
  + Funkcja up podnosi wartość semafora o jump i wartość raise o jump. Następnie gdy wartość raise osiągnęła co najmniej wartość jump pierwszej osoby w kolejce, jest ona usuwana z kolejki i jej wątek uruchamiany.
* Implementacja funkcji consume(person\*), produce(person\*):
  + Obie funkcje działają w pętli uzależnionej od semafora finish, opuszczanego po wpisaniu ZZ przez użytkownika do konsoli.
  + Konsument wykonuje w pętli kolejne operacje:
    - Down(full)
    - Down(mutex)
    - Remove\_item()
    - Up(mutex)
    - Up(empty)
    - Down(userMutex)
    - eatItem()
    - up(userMutex)
  + Producent wykonuje w pętli kolejne operacje:
    - Down(userMutex)
    - produceItem()
    - up(userMutex)
    - down(empty)
    - down(mutex)
    - enterItemToBuf()
    - up(mutex)
    - up(full)
* Symulacja równoległego wykonywania operacji przez użycie wątków z biblioteki <pthread.h>
* Kolejne operacje wykonywane w programie:
  + Definicja 4 struktur *person* i ich atrybutów
  + Definicja atrybutów semaforów
  + Uruchomienie 4 wątków, wykonujących funkcję consume/produce dla jednej z czterech struktur *person* (consumerA/consumerB/producerA/producerB)
  + Wątki współpracują z użytkownikiem:
    - Producenci pytają się, jakie litery wyprodukować
    - Konsumenci wyświetlają, jakie litery przeczytali
    - W przypadku wpisania ZZ, opuszczenie semafora *finish* i zakończenie wątków
  + Zapisywanie w dwóch plikach:
    - W pliku producer.txt liter wyprodukowanych przez producenta
    - W pliku consumer.txt liter skonsumowanych przez konsumenta
  + Po zakończeniu wszystkich wątków zakończenie działania programu.