*Kamil Bębenek*

# *SYSTEMY OPERACYJNE*

# *Rozwiązanie problemu synchronizacji procesów za pomocą semaforów na przykładzie problemu producentów--konsumentów*

## Opis

Program rozwiązuje problem producentów-konsumentów. Występują w nim dwa rodzaje procesów: producenci i konsumenci, które dzielą ze sobą bufor dla produkowanych i konsumowanych jednostek. Zadaniem producenta jest wytworzenie towaru i umieszczenie go w buforze. Konsument ma pobrać produkt z bufora.

Muszą być spełnione następujące założenia:

- Niedopuszczenie do czytania z pustego i zapisu do pełnego bufora.
- Procesy korzystające z bufora nie mogą ze sobą kolidować.

Dodatkowe założenia występujące w programie:

- Producenci produkują towary do momentu osiągnięcia wyznaczonego w programie limitu produktów, po czym kończą działanie.
- Konsumenci konsumują towary do momentu, w którym wszystkie produkty zostaną skonsumowane i kończą pracę, gdy nie będzie działał żaden producent.

W trakcie swojego działania program informuje na bieżąco o występujących zdarzeniach, takich jak:

- Rozpoczęcie i zakończenie działania procesu producenta lub konsumenta.
- Produkcja lub konsumpcja towaru.

## Przykład z życia

Wyobraźmy sobie bar szybkiej obsługi, w którym pracują kucharze, przygotowujący posiłki i umieszczający je w okienkach, oraz kelnerzy (pracownicy obsługujący klientów), którzy odbierają posiłki z okienka i dostarczają je odpowiednim klientom. Ważne jest, by kucharz nie przygotowywał kolejnych posiłków, gdy wszystkie okienka są zajęte, gdyż nie będzie miał gdzie ich położyć, z kolei kelner nie powinien tracić czasu na stanie przy okienkach, gdy nie będzie miał żadnego posiłku do odebrania.

## Uruchomienie programu

Uruchomienie programu z domyślnymi wartościami: ./synchronizacja

Uruchomienie programu z własnymi ustawieniami parametrów:

./synchronizacja -p 6 -k 5 -m 4 -n 2 -r 7 -i 20

## Domyślne wartości parametrów

- liczba producentów: 4 (parametr -p)
- liczba konsumentów: 3 (parametr -k)
- czas oczekiwania producentów: 5 (parametr -m)
- czas oczekiwania konsumentów: 3 (parametr -n)
- rozmiar bufora: 5 (parametr -r)
- limit wszystkich produktów: 16 (parametr -i)

## Wykaz plików

- argumenty.c - odpowiada za pobranie i ustawienie parametrów własnych,
    - argumenty.h - plik nagłówkowy,
- dokumentacja.pdf - dokumentacja programu,
- informacje.c - wyświetla użyte wartości parametrów,
    - informacje.h - plik nagłówkowy,
- konsument - plik wykonywalny, uruchamiający proces konsumenta,
- konsument.c - odpowiada za działanie procesów konsumenta,
- Makefile - plik reguł kompilacji,
- nadzorca_konsumentow.c - konstruowanie procesów oraz informowanie o zakończeniu pracy konsumentów,
    - nadzorca_konsumentow.h - plik nagłówkowy,
- nadzorca_producentow.c - konstruowanie procesów oraz informowanie o zakończeniu pracy producentów,
    - nadzorca_producentow.h - plik nagłówkowy,
- pamiec_dzielona.c - definicja struktury pamięci, parametry domyślne,
    - pamiec_dzielona.h - plik nagłówkowy,
- producent - plik wykonywalny, uruchamiający proces producenta,
- producent.c - odpowiada za działanie procesów producenta,
- semafory.c - inicjuje działanie semaforów,
    - semafory.h - plik nagłówkowy,
- synchronizacja - plik wykonywalny, uruchamiający docelowy program.

## Proponowane rozwiązanie

Procesy mają osobne sekcje danych - modyfikacje wykonane na danych w jednym procesie nie są dostępne dla drugiego. Aby procesy mogły mieć dostęp do tych samych danych należy:

- Skonstruować oddzielny segment pamięci.
- Udostępnić segment odpowiednim procesom.

Do skonstruowania pamięci dzielonej zostały wykorzystane wywołania systemowe w standardzie POSIX (pamiec_dzielona.c):

- shm_open() - skonstruowanie dzielonego segmentu pamięci.
- ftruncate() - ustalenie rozmiaru segmentu.
- mmap() - ustalenie odwzorowania segmentu w obszarze procesu.
- shm_unlink() - odłączenie lub skasowanie segmentu pamięci.

Do pamięci dzielonej inicjowane są domyślne wartości programu lub wartości przekazane w parametrach przez użytkownika (odpowiada za to plik argumenty.c przy użyciu funkcji getopt).

Aby zapewnić odpowiednią synchronizację działania producenta i konsumenta, oprócz przydzielania pamięci na bufor, należy utworzyć w systemie cztery semafory regulujące dostęp do sekcji krytycznej:

- pusty - przechowuje ilość pustych miejsc w kolejce (zapobiega zapisywaniu do pełnej kolejki).
- pelny - przechowuje ilość danych w kolejce (zapobiega czytaniu z pustego bufora).
- mutexK - semafor binarny blokujący dostępu do bufora dla więcej niż jednego procesu konsumenta.
- mutexP - semafor binarny blokujący dostępu do bufora dla więcej niż jednego procesu producenta.

Operacje wykonywane na semaforach (semafory.c):

- sem_init(S,1,N) - inicjacja semafora, ustawienie licznika semafora S na początkową wartość N w trybie dzielenia pamięci.
- sem_wait(S) - zajmowanie semafora, gdy licznik L semafora S jest dodatni (L > 0) zmniejsz go o 1 (L:=L-1). Gdy licznik L semafora S jest równy zero (L:=0) zablokuj proces bieżący.
- sem_post(S) - sygnalizacja semafora, gdy istnieje jakiś proces oczekujący na semaforze S to odblokuj jeden z czekających procesów. Gdy brak procesów oczekujących na semaforze S zwiększ jego licznik L o 1 (L:=L+1).
- sem_close(S) - zamknięcie semafora.

Korzystając z funkcji nadzorujących działanie procesów potomnych w plikach nadzorca_konsumentow.h oraz nadzorca_producentow.h, uruchamia się odpowiednio wiele procesów producentów (producent.c) oraz konsumentów (konsument.c). Wszystkie uruchomione procesy korzystają ze wspólnego uchwytu segmentu pamięci dzielonej. Dostęp do sekcji krytycznej reguluje semafor.

Gdy zostanie wyczerpany limit produktów do skonstruowania, procesy producentów kończą swoje działanie, a po skonsumowaniu wszystkich dostępnych produktów kończą działanie konsumenci. Na końcu działanie kończy program główny „synchronizacja”.

## Przykład działania programu

```
**********************************************************************************
Informacje o restauracji:
- liczba kucharzy: 4
- liczba kelnerów: 3
- czas przygotowania posiłku przez kucharza: 7
- czas dostarczenia posiłku do klienta przez kelnera: 3
- liczba okienek, w których kucharz umieszcza posiłki, a kelner je odbiera: 5
- maksymalna ilość posiłków przygotowywanych jednego dnia: 16
**********************************************************************************
Kucharz 5086 przyszedł do pracy.
Kucharz 5087 przyszedł do pracy.
Kucharz 5088 przyszedł do pracy.
Kucharz 5089 przyszedł do pracy.
Kelner 5090 przyszedł do pracy.
Kelner 5091 przyszedł do pracy.
Kelner 5092 przyszedł do pracy.
                                Kucharz 5089 przygotował nowy posiłek.
```

Kucharz 5088 przygotował nowy posiłek.
Kucharz 5087 przygotował nowy posiłek.
Kucharz 5086 przygotował nowy posiłek.
Kelner 5090 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 3.
Kelner 5091 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 2.
Kelner 5092 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 1.
Kelner 5090 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5089 przygotował nowy posiłek.
Kucharz 5088 przygotował nowy posiłek.
Kelner 5091 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 1.
Kelner 5092 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5087 przygotował nowy posiłek.
Kelner 5090 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5086 przygotował nowy posiłek.
Kelner 5091 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5089 przygotował nowy posiłek.
Kucharz 5088 przygotował nowy posiłek.
Kelner 5090 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 1.
Kelner 5091 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5087 przygotował nowy posiłek.
Kucharz 5086 przygotował nowy posiłek.
Kelner 5092 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 1.
Kelner 5090 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5089 przygotował nowy posiłek.
Kelner 5090 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5088 przygotował nowy posiłek.
Kelner 5092 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5087 przygotował nowy posiłek.
Kucharz 5086 przygotował nowy posiłek.
Kelner 5091 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 1.
Kelner 5090 zaniósł posiłek do klienta.
Pozostało posiłków w oknach: 0.
Kucharz 5086 zakończył pracę.
Kucharz 5087 zakończył pracę.
Kucharz 5088 zakończył pracę.
Kucharz 5089 zakończył pracę.
Kelner 5090 wyszedł z pracy.
Kelner 5091 wyszedł z pracy.

Kelner 5092 wyszedł z pracy.