Synchronizacja - semafory

Jakub Robaczewski

## Semafory:

Zaimplementuje 3 semafory:

* Queue\_full <- kolejka pełna
* Queue\_empty <- kolejka pusta
* Queue\_lock <- semafor mutex, zapewniający wyłączny dostęp

## Algorytm programu:

**Message:**

* Czekaj na podniesiony queue\_lock i podniesiony queue\_full, zablokuj queue\_lock
  + Można uśpić na losowy przedział czasu, jeśli zablokowany
* Wstaw element do kolejki (w przypadku VIP z przodu)
* queue\_empty += 1
* Sprawdz czy jest miejsce i ewentualnie queue\_full += 1
* Odblokuj queue\_lock
* Sprawdź czy osiągnął maksymalną liczbę wiadomości:
  + Zakończ program
* Kontynuuuj

**Main:**

* Jeżeli opuszczony semafor queue\_empty:
  + Czekaj na podniesienie semafora queue\_lock, zablokuj
  + Wyciągnij element z kolejki
  + queue\_full -= 1
  + queue\_empty -= 1
  + Odblokuj semafor queue\_lock
* Po obsłużeniu określonej liczby sygnałów – zakończ działanie

**Główna pętla:**

Uruchom main

Uruchom 2 instancje message (normalne i VIP), realizowane za pomocą fork() lub pthread

## Testowanie:

* Wypisywanie czasu od początku działania programu (time(0)) dla każdej wypisywanej wiadomości. Ułatwi sprawdzanie kolejności
* Każda zmiana stanu semafora printuje jego stan na ekranie.
* Log jest również zapisywany do pliku, żeby ułatwić jego późniejszą analizę
* Wiadomości generowane przez pisarzy są w postaci {id\_wiadomości}\_{id\_pisarza}