Przetwarzanie współbieżne

Laboratorium 3

Jan Bartula

# Zadanie 1

## Opis

Aby program działał poprawnie należy skorzystać z elementów chroniących dostęp do danych współdzielonych. W naszym przypadku jest to zmienna, do której mają dostęp dwa wątki.

Należy więc wykonać zablokowanie t.zw. mutexa przed dostępem do zmiennej, a następnie go odblokować po zakończonej pracy.

Zadanie można wykonać w sposób dwojaki:

* Zablokować dostęp w jednej funkcji po zakończeniu dodawania odblokować, żeby druga mogła zacząć odejmować, co uśpi jeden z wątków na czas operacji pierwszego
* Blokować i odblokowywać iteracyjnie w czasie każdej operacji. Dzięki czemu oba wątki będą w *założeniu*naprzemiennie inkrementować i dekrementować

Pierwsze rozwiązanie będzie szybsze, ponieważ blokada i zwolnienie wystąpi tylko raz, natomiast iteracyjne blokowanie będzie naprzemiennie zwalniać i odblokowywać dostęp do zmiennej, co wykona się tyle samo razy co liczba powtórzeń w pętli.

Sumarycznie i tak w obu przypadkach wynik będzie taki sam.

## Implementacja

void \*increment(void \*arg) {  
 int r, i, tmp;  
 for (i = 0; i < 100000; i++) {  
 // oczekiwanie na możliwość zablokowania  
 if ((r = pthread\_mutex\_lock(&mutex)) != 0) {  
 printError(r);  
 exit(1);  
 }  
 tmp = global\_variable;  
 tmp++;  
 printf("%d\n", tmp);  
 global\_variable = tmp;

// zwolnienie blokady  
 if ((r = pthread\_mutex\_unlock(&mutex)) != 0) {  
 printError(r);  
 exit(1);  
 }  
 }  
  
 return 0;  
}

// druga funkcja analogicznie ale z dekrementacją

# Zadanie 2

Symulator baru został zaimplementowany następująco, jest:

* Mutexa kelnera – który sprawdza, czy są wolne miejsca w barze i czy jest jeszcze możliwość wpuszczania klientów (czy jest po 6:00 a przed 22:00)
* Mutex na każde piwo – ponieważ każde piwo ma być unikalne (teoretycznie kelner mógłby też je wydawać, ale wtedy nie wiadomo kto ma jakie piwo)
* Klienci wchodząc do baru czekają maksymalnie 1h (1 sekundę w rzeczywistości) aż kelner ich wpuści, w przeciwnym wypadku wątek się kończy.
* Następnie namyślają się czy chcą piwo.
* Czekają na 1 piwo, wypijają je w losowym czasie, oddają piwo i czekają na drugie piwo.   
  Po drugim piwie opuszczają bar zwalniając krzesło
* Krzeseł w barze jest LICZBA\_PIW + 5
* Klienci są liniowo rozdysponowani pomiędzy 00:00 a 02:00 następnego dnia
* Symulacja zaczyna się o godzinie 00:00 dnia pierwszego

## Przed 6:00

Tales 👩 id: 12, i wait one hour, bye! (bar closed)

Time: 4:00

Tales 👩 id: 16, enter bar.

Patryk 🙍 id: 13, i wait one hour, bye! (bar closed)

Oskar 👶 id: 17, enter bar.

Platon 👳 id: 14, i wait one hour, bye! (bar closed)

Herakles 🤶 id: 18, enter bar.

Oskar 👶 id: 15, i wait one hour, bye! (bar closed)

Oskar 👶 id: 19, enter bar.

Tales 👩 id: 16, i wait one hour, bye! (bar closed)

## Po 6:00

Patryk 🙍 id: 23, enter bar.

Pitagoras 🧓 id: 20, has a seat.

Time: 6:00

Tales 👩 id: 24, enter bar.

Tales 👩 id: 24, has a seat.

Dawid 👴 id: 21, has a seat.

Jakub 👲 id: 25, enter bar.

Jakub 👲 id: 25, has a seat.

Jakub 👲 id: 22, has a seat.

Herakles 🤶 id: 26, enter bar.

Herakles 🤶 id: 26, has a seat.

Patryk 🙍 id: 23, has a seat.

Time: 7:00

Patryk 🙍 id: 27, enter bar.

## Picie piwa

Filip 👨 id: 30, has a seat.

Pitagoras 🧓 id: 28, is drinking 1 🍺:7 now

Dawid 👴 id: 21, released 1 🍺:0 after 893ms.

Dawid 👴 id: 21, is drinking 2 🍺:0 now

Herakles 🤶 id: 26, released 1 🍺:2 after 729ms.

Herakles 🤶 id: 26, is drinking 2 🍺:2 now

Platon 👳 id: 68, is full 🤮

## Po 22:00

Pitagoras 🧓 id: 78, released 2 🍺:4 after 536ms.

Pitagoras 🧓 id: 78, is full 🤮

Pitagoras 🧓 id: 78, left bar

Jakub 👲 id: 85, i wait one hour, bye! (bar closed)

Time: 23:00

Pitagoras 🧓 id: 89, enter bar.

Dawid 👴 id: 84, is drinking 1 🍺:2 now

Herakles 🤶 id: 86, i wait one hour, bye! (bar closed)

Patryk 🙍 id: 75, released 2 🍺:0 after 1124ms.

## Symulacja czeka do ostatniego klienta i kończy się:

Time: 25:00

Platon 👳 id: 97, enter bar.

Patryk 🙍 id: 94, i wait one hour, bye! (bar closed)

Dawid 👴 id: 98, enter bar.

Patryk 🙍 id: 95, i wait one hour, bye! (bar closed)

Tales 👩 id: 99, enter bar.

Platon 👳 id: 96, i wait one hour, bye! (bar closed)

Dawid 👴 id: 84, released 2 🍺:1 after 1496ms.

Dawid 👴 id: 84, is full 🤮

Dawid 👴 id: 84, left bar

Platon 👳 id: 97, i wait one hour, bye! (bar closed)

Dawid 👴 id: 98, i wait one hour, bye! (bar closed)

Tales 👩 id: 99, i wait one hour, bye! (bar closed)

Thank you for attention.

Czasami logi mogą się nie zgadzać kolejnością, ponieważ aby konsola była czysta, zastosowałem także mutexa na dostęp do pisania na konsoli, przez co w zależności kto się dostanie pierwszy, ten pierwszy coś napisze.