Programowanie równoległe Informatyka Stosowana

Laboratorium 4

Cel:

opanowanie umiejętności pisania programów z synchronizacją wątków.

Kroki:

1. Utworzenie katalogu roboczego (np. lab_4)
2. Zaprojektowanie symulacji pubu z następującą specyfikacją:
    1. W pubie jest n kufli 1 L oraz k klientów.
    2. Klienci są reprezentowani przez wątki.
    3. Każdy klient pragnie wypić m litrów piwa.
    4. W pubie czekają przygotowane puste kufle do pobrania przez klientów.
    5. Napełnienie jednego kufla trwa kilka sekund (jest jeden kran!).
    6. Klient opróżnia kufel w kilkanaście sekund.
    7. Po wypiciu każdego litra klient oddaje stary kufel i pobiera nowy
    8. Po wypiciu m litrów klient opuszcza pub.
    9. Pub otwarty jest do ostatniego klienta (ale nie wpuszcza nowych).
    10. Każdy klient podczas pobytu w pubie informuje (wypisując na ekranie) co robi w danej chwili
3. W pierwszej wersji należy zaprojektować i zaimplementować symulację pubu używając wyłącznie muteksów i procedury pthread_mutex_lock (sekundy można zamienić na mikrosekundy – usleep lub nanosleep zamiast sleep – dla przyspieszenia testowania programu). Parametrami symulacji są: liczba klientów oraz liczba kufli.

    Wskazówka: rozpocząć od sytuacji kiedy liczba kufli przewyższa liczbę klientów i od rozwiązania problemu – jak ma zachowywać się klient (czyli nowo utworzony wątek), żeby sprawdzać dostępność kufli i pobrać wolny kufel (rozwiązanie problemu wymaga, aby najpierw rozstrzygnąć jak maja być reprezentowane kufle).

    Punktem wyjścia powinien być program pub_sym_1.c . Program nie jest poprawny ponieważ w momencie kiedy klient pobiera kufel nie wiadomo czy jest jakiś kufel wolny. Reprezentacja kufli musi umożliwiać sprawdzenie czy jest jakiś kufel wolny przed pobraniem kufla (klienci powinni także wyświetlać informację wskazującą na to czy są wolne kufle).
4. W drugiej kolejności rozważyć sytuację kiedy liczba klientów jest większa niż liczba kufli. Jeśli korzystamy tylko z muteksów musimy zastosować aktywne czekanie, busy waiting (po nieudanej próbie pobrania kufla klient odchodzi od kranu i może robić coś innego). Można wykorzystać wzór (kropki oznaczają miejsca, gdzie musi zostać coś uzupełnione):

```
int sukces = 0;
do{
  .....
  if ( can_continue ) { .......; sukces = 1; }
  .....
  if ( sukces == 0 ) do_something_else_or_nothing(); // can be omitted
  else{ ..... }
} while ( sukces == 0 );
```

Rozszerzenia dla podwyższenia oceny:

1. W związku z rosnącą popularnością pubu (rosnąca liczba klientów) napisanie nowego programu (np. na podstawie dostarczonego szkieletu pub_sym_2.c) - wprowadzenie większej liczby kufli i kranów oraz procedury trylock do efektywnej obsługi możliwości korzystania z wielu kranów i kufli. Należy również zastosować aktywne czekanie, ale teraz krany maja być rozróżnialne, co oznacza, że ich reprezentacja w programie musi się zmienić.

2. Sprawdzenie wykorzystania zasobów systemu w trakcie działania programu (poprzez uruchomienie dowolnego programu śledzącego – np. top, htop lub graficznego).

3. Zilustrowanie graficzne działania pubu.

Warunki zaliczenia:

Obecność na zajęciach i wykonanie kroków 1-4.

Oddanie napisanego odręcznie sprawozdania z opisem zadania, kodem źródłowym programów oraz analizą wydajności dla różnych parametrów symulacji.