|  |  |  |
| --- | --- | --- |
| Łukasz Oprych  Gr.5 ITE | Programowanie Równoległe | Laboratorium 4, Wzajemne wykluczanie |

**Cel ćwiczenia:**

Celem zadania było poznanie różnych technik synchronizacji wątków podczas korzystania z wspólnych zasobów w taki sposób, aby uniknąć wystąpienia sytuacji, w której doszłoby do wyścigu danych.

**Przebieg ćwiczenia:**

Po utworzeniu katalogu lab\_4 oraz skonfigurowaniu programu do pracy, uzupełniono kod pub\_sym\_1.c o śledzenie liczby kufli w funkcji wątek\_klient oraz wyświetlanie końcowej liczby w funkcji main.

**Kod:**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Wynik symulacji:**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Jak widać dochodzi do błędu, liczba kufli się zwiększyła, nie jest ona zgodna z wartością początkową.

Aby zapobiec temu problemowi dokonano zabezpieczenia dostępu do zasobu za pomocą funkcji pthread\_mutex\_lock oraz pthread\_mutex\_unlock przy zliczaniu kufli oraz przy korzystaniu z kranu.

**Kod:**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

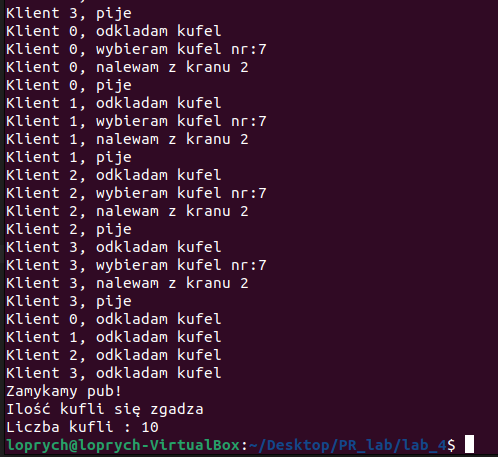
Opis wygenerowany automatycznie**

Przy wykonywaniu tej części dodano również zmienną globalną l\_kranów, której wartość przyrównano w main z l\_kr. Na przykładzie kufli: W każdym wątku, przed pobraniem kufla, zmniejszano wartość zmiennej globalnej, która trzymała liczbę dostępnych kufli. To odbywało się po zablokowaniu muteksu za pomocą pthread\_mutex\_lock. W sekcji krytycznej pobierano kufel i odblokowywano mutex. Analogicznie, zwracanie kufli odbywało się poprzez zwiększanie tej samej zmiennej globalnej. Na zakończenie działania programu dodano wyświetlanie wartości zmiennej liczba\_kufli w celu sprawdzenia, czy końcowa wartość zmiennej jest równa początkowej. Ten mechanizm służył do określenia, czy operacje na kuflach odbywały się bez wyścigu.

**Wynik symulacji:**

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**Ilość klientów < ilość kufli:

****

Ilość klientów > ilość kufli:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Jak widać, przy symulacji ilość kufli zgadza się na koniec, lecz w przypadku symulacji z większą ilością klientów, wykorzystywany jest nieistniejący kufel -1, ponieważ skończyły się w trakcie.

Kolejnym zadaniem było dokonanie symulacji z uwzględnieniem aktywnego czekania na zasób. Z tego powodu utworzono plik pub\_sym\_1\_2.c

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie**Kod:**

W tym wypadku wykorzystano pętlę do while oraz zmienną success, aby klient mógł pobrać kufel tylko wtedy, gdy jest dostępny i istnieje, success = 1, w innym wypadku klient musi czekać na zwolnienie kufla.

**Wynik Symulacji:**

**Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie**

**Wnioski:**

Aby zapewnić poprawne współdzielenie zasobów pomiędzy wątkami, wymagane jest zapewnienie synchronizacji. Najprościej w takim przypadku zastosować mutex, lecz nie w nadmiernych ilościach, aby nie blokować wątków przez czekanie na zamknięcie mutexa. Mutex pozwala na zarządzanie dostępem do sekcji krytycznej w taki sposób, aby dany fragment kodu nie został wykonany jednocześnie. W celu zniwelowania wad użycia tylko mutexów, w tym przypadku pętla do while zapewnia poprawność wykonywania działań, zarządzając dostępem do kufli.