Łukasz Oprych	Programowanie Równoległe	Laboratorium 4, Wzajemne
Gr.5 ITE		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:

```
<mark>if</mark> (liczba_kufli == l_kf ) {
 printf("\nIlość kufli się zgadza\n");
 printf("\nLiczba kufli się nie zgadza\n");
 printf("\nLiczba kufli: %d", liczba_kufli);
void * watek klient (void * arg wsk){
 int moj_id = * ((int *)arg_wsk);
 int i, j, kufel, result;
 int ile_musze_wypic = ILE_MUSZE_WYPIC;
 long int wykonana praca = 0;
 //printf("\nKlient %d, wchodze do pubu\n", moj_id);
 for(i=0; i<ile_musze_wypic; i++){</pre>
    tf (liczba_kufli > 0){
    //printf("\nKlient %d, wybieram kufel\n", moj id);
    liczba_kufli--;
    wykonana praca++;
   //printf("\nKlient %d, nalewam z kranu %d\n", moj_id, j);
    usleep(1);
    //printf("\nKlient %d, pije\n", moj_id);
    //nanosleep((struct timespec[]){{0, 50000000L}}, NULL);
    liczba_kufli++;
```

Wynik symulacji:

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_4$ ./pub_sym_1
Liczba klientow: 4
Liczba kufli: 2
Liczba kranow: 1
Otwieramy pub (simple)!
Liczba wolnych kufli 2
Zamykamy pub!
Liczba kufli się nie zgadza
Liczba kufli: 4
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_4$
```

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:

```
void * watek_klient (void * arg_wsk){
  int moj_id = * ((int *)arg_wsk);
  int ile_musze_wypic = ILE_MUSZE_WYPIC;
  long int wykonana_praca = 0;
  printf("\nKlient %d, wchodze do pubu", moj_id);
  for(i=0; i<ile_musze_wypic; i++){</pre>
    pthread_mutex_lock (&mutex);
    printf("\nKlient %d, wybieram kufel nr:%d", moj_id, liczba_kufli);
liczba_kufli--;
    pthread_mutex_unlock (&mutex);
     //i=0:
    pthread_mutex_lock (&mutex);
    printf("\nKlient %d, nalewam z kranu %d", moj_id, l_kranow);
usleep(1);
    pthread_mutex_unlock (&mutex);
    printf("\nKlient %d, pije", moj_id);
nanosleep((struct timespec[]){{0, 500000000L}}, NULL);
pthread_mutex_lock (&mutex);
printf("\nKlient %d, odkladam kufel", moj_id);
liczba_kufli++;
    pthread_mutex_unlock (&mutex);
  return(NULL);
```

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:

Ilość klientów < ilość kufli:

```
oprych@loprych-VirtualBox:
                                         lab/lab_4$ ./pub_sym_1
                                                                  Klient 3, pije
                                                                  Klient O, odkladam kufel
Liczba klientow: 4
                                                                  Klient 0, wybieram kufel nr:7
Liczba kufli: 10
                                                                  Klient 0, nalewam z kranu 2
                                                                  Klient 0, pije
Otwieramy pub (simple)!
                                                                  Klient 1, odkladam kufel
                                                                  Klient 1, wybieram kufel nr:7
Liczba wolnych kufli 10
                                                                  Klient 1,
                                                                              nalewam z kranu 2
                                                                  Klient 1, pije
Klient 0, wchodzę do pubu
Klient 0, wybieram kufel nr:10
                                                                  Klient 2, odkladam kufel
Klient 0, nálewam z kranu 2
                                                                  Klient 2, wybieram kufel nr:7
Klient 0, pije
                                                                  Klient 2, nalewam z kranu 2
Klient 1, wchodzę do pubu
Klient 1, wybieram kufel nr:9
Klient 1, nalewam z kranu 2
                                                                  Klient 2, pije
                                                                  Klient 3, odkladam kufel
                                                                  Klient 3, wybieram kufel nr:7
Klient 1, pije
Klient 2,
Klient 2,
          wchodzę do pubu
                                                                  Klient 3, nalewam z kranu 2
          wybieram kufel nr:8
                                                                  Klient 3, pije
Klient
         nalewam z kranu 2
                                                                  Klient 0, odkladam kufel
Klient 3,
          wchodzę do pubu
                                                                  Klient 1, odkladam kufel
Klient 2,
         pije
                                                                  Klient 2, odkladam kufel
Klient
          wybieram kufel nr:7
                                                                  Klient 3, odkladam kufel
Klient
          nalewam z kranu 2
                                                                  Zamykamy pub!
Ilość kufli się zgadza
          pije
odkladam kufel
wybieram kufel nr:7
Klient 3.
Klient 0.
Klient 0,
                                                                  Liczba kufli : 10
Klient 0,
          nalewam z kranu 2
                                                                   loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_4$
```

Ilość klientów > ilość kufli:

```
4$ ./pub sym 1
                                                                                         Klient 4, wybieram kufel nr:-1
Liczba klientow: 10
                                                                                         Klient 4, nalewam z kranu 1
                                                                                         Klient 4, pije
Liczba kufli: 4
                                                                                          (lient 3, odkladam kufel
Otwieramy pub (simple)!
                                                                                         Klient 5, odkladam kufel
Liczba wolnych kufli 4
                                                                                         Klient 5, wybieram kufel nr:0
                                                                                         Klient 5, nalewam z kranu 1
Klient 0, wchodzę do pubu
Klient 0, wybieram kufel nr:4
Klient 1, mchodzę do pubu
Klient 1, wchodzę do pubu
Klient 3, wchodzę do pubu
Klient 3, wchodzę do pubu
Klient 4, wchodzę do pubu
Klient 6, wjie
Klient 6, wchodzę do pubu
Klient 6, wchodzę do pubu
Klient 6, wzbieram kufel nr:3
Klient 6, wgbieram kufel nr:3
                                                                                         Klient 5, pije
                                                                                         (lient 7, odkladam kufel
                                                                                         Klient 9, odkladam kufel
                                                                                         Klient 1, odkladam kufel
                                                                                         Klient 4, odkladam kufel
                                                                                         Klient 5, odkladam kufel
Klient 6, wpieram kurel nr
Klient 6, nalewam z kranu 1
Klient 5, wchodzę do pubu
Klient 7, wchodzę do pubu
Klient 8, wchodzę do pubu
Klient 9, wchodzę do pubu
                                                                                         Zamykamy pub!
                                                                                          lość kufli się zgadza
                                                                                          iczba kufli : 4
                                                                                         loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_4$
Klient 6, pije
Klient 2, wybieram kufel nr:2
Klient 2, nalewam z kranu 1
Klient 2, pije
Klient 4, wybieram kufel nr:1
Klient 4, nalewam z kranu 1
Klient 4,́ pije
Klient 5, wybieram kufel nr:0
Klient 5, nalewam z kranu 1
Klient 5, pije
 (lient 8. wybieram kufel nr:-1
         8, nalewam z kranu
8, pije
```

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

Kod:

```
for(i=0; i<ile_musze_wypic; i++){
  int success = 0;

do {
  int mutex_locked = 0;
  pthread_mutex_lock (&mutex);
  if (mutex_locked == 0){
    if (liczba_kufli > 0){
    success = 1; // opuszczenie pętli
    printf("\nklient %d, wybieram kufel nr: %d", moj_id, liczba_kufli);
    liczba_kufli--;
  }
  pthread_mutex_unlock (&mutex);
  }
  //wykonana_praca++;
  }

while (success == 0);
  pthread_mutex_lock (&mutex);
  printf("\nklient %d, nalewam z kranu %d", moj_id, l_kranow);
  usleep(1);

printf("\nklient %d, pije", moj_id);
  pthread_mutex_unlock (&mutex);
  nanosleep((struct timespec[]){{0, 500000000L}}, NULL);
  pthread_mutex_lock (&mutex);
  liczba_kufli++;
  printf("\nklient %d, odkladam kufel, wybieram nr: %d", moj_id, liczba_kufli);
  pthread_mutex_unlock (&mutex);
}
```

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:

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_4/2$ ./pub_sym_1_2
Liczba klientow: 5
Liczba kufli: 10
Otwieramy pub (simple)!
Liczba wolnych kufli 10
Klient 0, wchodzę do pubu
Klient 0, wybieram kufel nr: 10
Klient 0, nalewam z kranu 0
Klient 1, wchodzę do pubu
Klient 2, wchodzę do pubu
Klient 0, pije
Klient 1, wybieram kufel nr: 9
Klient 1, nalewam z kranu 0
Klient 3, wchodzę do pubu
Klient 4, wchodzę do pubu
Klient 1, pije
Klient 3, wybieram kufel nr: 8
Klient 3, nalewam z kranu 0
Klient 3, pije
Klient 4, wybieram kufel nr: 7
Klient 4, nalewam z kranu 0
Klient 4, pije
Klient 2, wybieram kufel nr: 6
Klient 2, nalewam z kranu 0
Klient 2, pije
```

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.