Przemysław Szymoniak

Przetwarzanie współbieżne. Programowanie równoległe i rozproszone.

Sprawozdanie z laboratorium 4.

Celem laboratorium było nabycie praktycznej umiejętności pisania programów z synchronizacją wątków.

W ramach zajęć zrealizowałem następujące kroki:

* Pobranie plików „**pub\_sym\_1.c**” oraz **„pub\_sym\_2.c”** i skopiowanie ich do wcześniej utworzonego folderu
* Zaprojektowanie symulacji pubu w sytuacji gdy liczba kufli przewyższa liczbę klientów wraz z jego implementacją w pliku **„pub\_sym\_1.c”:**

|  |  |
| --- | --- |
| **void \* watek\_klient (void \* arg);**  **int l\_kf;**  **pthread\_mutex\_t**  **mutex=PTHREAD\_MUTEX\_INITIALIZER;**  **main(){**  **pthread\_t \*tab\_klient;**  **int \*tab\_klient\_id;**  **int l\_kl, l\_kr, i;**  **l\_kr = 1;**  **printf("\nLiczba klientow: "); scanf("%d", &l\_kl);**  **printf("\nLiczba kufli: "); scanf("%d", &l\_kf);**  **tab\_klient= (pthread\_t \*) malloc(l\_kl\*sizeof(pthread\_t));**  **tab\_klient\_id = (int \*) malloc(l\_kl\*sizeof(int));**  **for(i=0;i<l\_kl;i++)**  **tab\_klient\_id[i]=i;**  **printf("\nOtwieramy pub (simple)!\n");**  **printf("\nLiczba wolnych kufli %d\n", l\_kf);**  **for(i=0;i<l\_kl;i++){**  **pthread\_create(&tab\_klient[i], NULL, watek\_klient, &tab\_klient\_id[i]);**  **}**  **for(i=0;i<l\_kl;i++){**  **pthread\_join( tab\_klient[i], NULL);**  **}**  **printf("\nZamykamy pub!\n");**  **}** | **void \* watek\_klient (void \* arg\_wsk){**  **int moj\_id = \* ((int \*)arg\_wsk);**  **int i, j=0;**  **int ile\_musze\_wypic = 2;**  **printf("\nKlient %d, wchodzę do pubu\n", moj\_id);**    **for(i=0; i<ile\_musze\_wypic; i++){**  **pthread\_mutex\_lock(&mutex);**  **if(l\_kf > 0){**  **l\_kf--;**  **printf("\nKlient %d, wybieram kufel %d\n", moj\_id), j;**  **printf("\nKlient %d, nalewam z kranu %d\n", moj\_id, j);**  **usleep(300);**  **pthread\_mutex\_unlock(&mutex);**    **printf("\nKlient %d, pije\n", moj\_id);**  **nanosleep((struct timespec[]){{0, 5000000L}}, NULL);**  **pthread\_mutex\_lock(&mutex);**  **printf("\nKlient %d, odkladam kufel\n", moj\_id);**  **l\_kf++;**  **pthread\_mutex\_unlock(&mutex);**  **}**  **}**  **printf("\nKlient %d, wychodzę z pubu\n", moj\_id);**  **return(NULL);**  **}** |

* Na podstawie pliku **„pub\_sym\_1.c”** wykonanie programu z uwzględnieniem sytuacji gdy liczba kufli jest większa od liczby klientów, dokonanie zmiany (w funkcji wątku) to:
  + Wykorzystanie aktywnego czekania
  + Obsługa braku obecności kufla
  + W wypadku braku kufla wymuszenie odczekania chwili
  + Podział na część kodu odpowiedzialna za odbieranie kufla oraz za picie i odłożenie go
  + Dodanie komunikatów odnośnie liczby dostępnych kufli
  + **ZMODYFIKOWANA CZĘŚĆ PROGRAMU:**

**do{**

**pthread\_mutex\_lock(&mutex);**

**if(l\_kf > 0){**

**sucess = 1;**

**l\_kf--;**

**printf("\nKlient %d, wybieram kufel %d\n", moj\_id, j);**

**printf("\nKlient %d, nalewam z kranu %d\n", moj\_id, j);**

**usleep(300);**

**pthread\_mutex\_unlock(&mutex);**

**}else{**

**printf("Klient %d, brak wolnych kufli\n", moj\_id);**

**pthread\_mutex\_unlock(&mutex);**

**usleep(200);**

**}**

**}while(!sucess);**

**printf("\nKlient %d, pije\n", moj\_id);**

**nanosleep((struct timespec[]){{0, 5000000L}}, NULL);**

**pthread\_mutex\_lock(&mutex);**

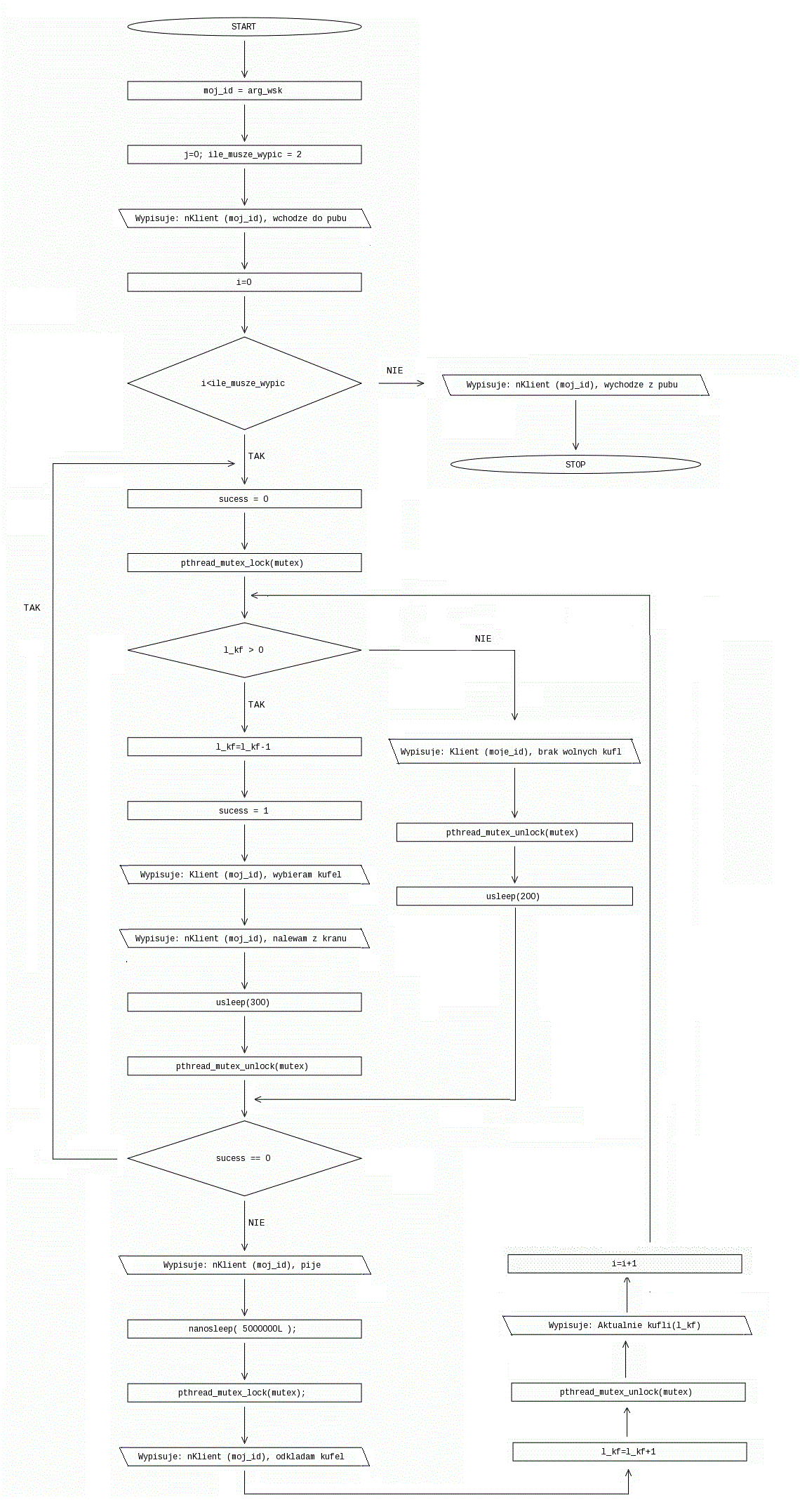
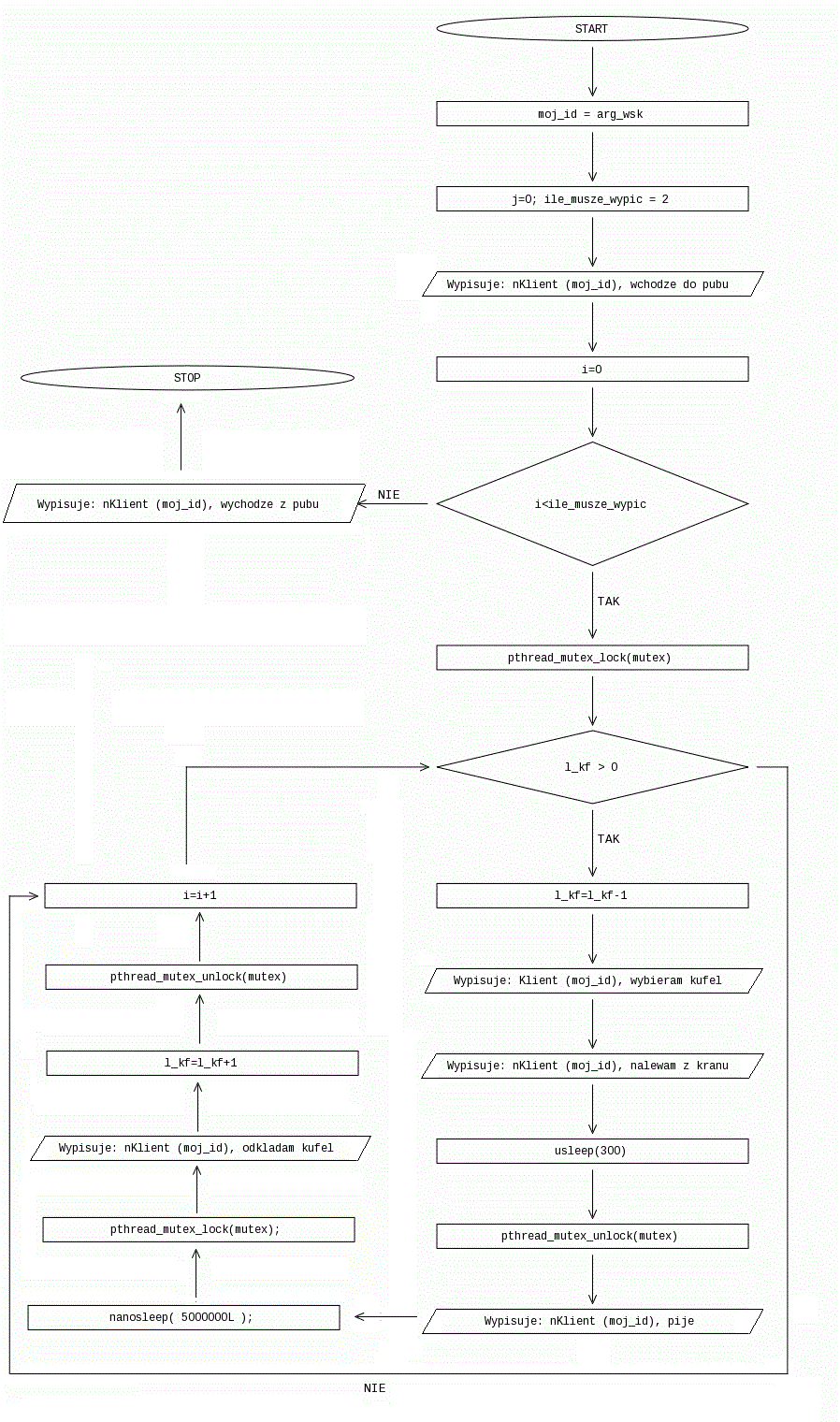
**printf("\nKlient %d, odkladam kufel\n", moj\_id);**

**l\_kf++;**

**pthread\_mutex\_unlock(&mutex);**

**printf("\nAktualnie kufli:%d\n", l\_kf);**

* Schemat blokowy programu (W większej skali dołączone jako załącznik):



Wnioski:

* Wykorzystanie mutexów pozwoliło znacząco zwiększyć bezpieczeństwo pracy aplikacji, poprzez blokowanie innych wątków chcących dostać się równocześnie do współdzielonych danych
* Użycie algorytmu z aktywnym czekaniem pozwoliło zmniejszyć czas blokowania wątków pozwalając im wykonywać w między czasie inne operacje nie wymagające dostępu do danych współdzielonych
* Zaimplementowane rozwiązanie w działało poprawnie przy każdym testowanym wariancie
* Większa liczba kranów wymagałaby od programu użycia tablicy mutexów oraz kranów
* Blokada mutexem została wykorzystana osobno przy odbieraniu i oddawaniu kufla aby inne wątki mogły wykorzystać ten czas do pracy równoległej