Tomasz Ligęza

**Programowanie równoległe. Przetwarzanie równoległe i rozproszone.**

Sprawozdanie z laboratorium 8.

Cel zajęć:

* doskonalenie umiejętności realizacji synchronizacji w języku C za pomocą zmiennych warunku oraz w programach obiektowych w Javie za pomocą narzędzi pakietu java.util.concurrency.

W ramach zajęć zrealizowałem następujące kroki:

1. Utworzyłem katalog roboczy lab\_8,
2. Przetestowałem działanie programu bariera, bez żadnych zmian w kodzie.
3. Wprowadziłem zmiany w kodzie pliku bariera.c tak, by program działał poprawnie:
   1. dodałem zmienne globalne typu int: liczba\_watkow\_aktualna, liczba\_watkow\_docelowa,
   2. dodałem zmienne globalne muteksu oraz warunku,
   3. aby zapewnić poprawność przy wielokrotnym uruchamianiu funkcji bariera należy zerować zmienną liczba\_watkow\_aktualna gdy jest ona równa liczba\_watkow\_docelowa,

int liczba\_watkow\_aktualna;

int liczba\_watkow\_docelowa;

pthread\_mutex\_t muteks = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_t warunek;

void bariera\_init(int liczba\_watkow) {

liczba\_watkow\_aktualna = 0;

liczba\_watkow\_docelowa = liczba\_watkow;

pthread\_mutex\_init(&muteks, NULL);

pthread\_cond\_init(&warunek, NULL);

}

void bariera() {

pthread\_mutex\_lock(&muteks);

liczba\_watkow\_aktualna++;

if(liczba\_watkow\_aktualna != liczba\_watkow\_docelowa) {

pthread\_cond\_wait(&warunek, &muteks);

}

else {

liczba\_watkow\_aktualna = 0;

pthread\_cond\_broadcast(&warunek);

}

pthread\_mutex\_unlock(&muteks);

}

1. Pobrałem paczkę CzytPis\_Pthreads.tgz, rozpakowałem ją oraz uruchomiłem program, aby zidentyfikować problem.
2. Uzupełniłem pola struktury czytelnia\_t:

/\*\*\* Definicje typow zmiennych \*\*\*/

typedef struct {

int liczba\_czyt;

int liczba\_pis;

int liczba\_czekajacych\_pisarzy;

int liczba\_czekajacych\_czytelnikow;

pthread\_cond\_t pisarze;

pthread\_cond\_t czytelnicy;

pthread\_mutex\_t muteks;

// <- zasoby czytelni

} czytelnia\_t;

1. Dodałem sprawdzanie warunków poprawnych wartości aktualnych liczb pisarzy i czytelników w procedurach pisania i czytania:

void czytam(czytelnia\_t \*czytelnia\_p) {

usleep(rand() % 300000);

if(czytelnia\_p->liczba\_pis > 1 ||

(czytelnia\_p->liczba\_pis == 1 && czytelnia\_p->liczba\_czyt > 0)) {

printf("Blad! piszacych: %d, czytajacych: %d.",

czytelnia\_p->liczba\_pis, czytelnia\_p->liczba\_czyt);

exit(1);

}

}

//w metodzie warunek wygląda identycznie

1. Na podstawie pseudokodu monitora Czytelnia poprawiłem kod tak by rozwiązać problem czytelników i pisarzy z użyciem zmiennych warunku:

int my\_read\_lock\_lock(czytelnia\_t \*czytelnia\_p) {

pthread\_mutex\_lock(&czytelnia\_p->muteks);

if(czytelnia\_p->liczba\_pis > 0 ||

czytelnia\_p->liczba\_czekajacych\_pisarzy > 0) {

czytelnia\_p->liczba\_czekajacych\_czytelnikow++;

pthread\_cond\_wait(&czytelnia\_p->czytelnicy, &czytelnia\_p->muteks);

czytelnia\_p->liczba\_czekajacych\_czytelnikow--;

}

czytelnia\_p->liczba\_czyt++;

pthread\_mutex\_unlock(&czytelnia\_p->muteks);

pthread\_cond\_signal(&czytelnia\_p->czytelnicy);

}

int my\_read\_lock\_unlock(czytelnia\_t \*czytelnia\_p) {

pthread\_mutex\_lock(&czytelnia\_p->muteks);

czytelnia\_p->liczba\_czyt--;

if(czytelnia\_p->liczba\_czyt == 0)

pthread\_cond\_signal(&czytelnia\_p->pisarze);

pthread\_mutex\_unlock(&czytelnia\_p->muteks);

}

int my\_write\_lock\_lock(czytelnia\_t \*czytelnia\_p) {

pthread\_mutex\_lock(&czytelnia\_p->muteks);

if(czytelnia\_p->liczba\_pis + czytelnia\_p->liczba\_czyt > 0) {

czytelnia\_p->liczba\_czekajacych\_pisarzy++;

pthread\_cond\_wait(&czytelnia\_p->pisarze, &czytelnia\_p->muteks);

czytelnia\_p->liczba\_czekajacych\_pisarzy--;

}

czytelnia\_p->liczba\_pis++;

pthread\_mutex\_unlock(&czytelnia\_p->muteks);

}

int my\_write\_lock\_unlock(czytelnia\_t \*czytelnia\_p) {

pthread\_mutex\_lock(&czytelnia\_p->muteks);

czytelnia\_p->liczba\_pis--;

if(czytelnia\_p->liczba\_czekajacych\_czytelnikow > 0)

pthread\_cond\_signal(&czytelnia\_p->czytelnicy);

else

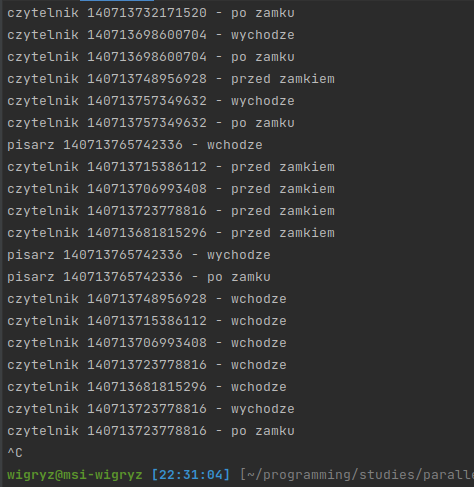
pthread\_cond\_signal(&czytelnia\_p->pisarze);

pthread\_mutex\_unlock(&czytelnia\_p->muteks);

}

1. Uruchomiłem program i po kilku minutach poprawnego działania uznałem, że zrealizowałem zadanie prawidłowo.

Zrzut ekranu z wykonania programu:



Wnioski:

Pierwszym problemem był problem bariery - wątek może zakończyć pewną czynność dopiero, gdy reszta wątków zacznie wykonywać tę czynność. Do rozwiązania tego problemu wykorzystaliśmy zmienną warunku. W momencie, gdy wątek wchodził do funkcji i nie był on “ostatnim” wątkiem, to przechodził w stan oczekiwania. Gdy jednak do funkcji wchodził wątek ostatni, to wywoływał on funkcję pthread\_cond\_broadcast ze zmienną warunku jako argumentem. W ten sposób informował on pozostałe oczekujące wątki o możliwości kontynuacji pracy.  
 Problem pisarzy i czytelników to często występujący schemat - grupa procesów modyfikuje zasób, podczas gdy druga grupa jedynie go odczytuje. W celu rozwiązania tego problemu wykorzystuje się zmienne warunku. Dzięki nim możemy w łatwy sposób (wcale nie taki łatwy ;)) zarządzać współdzielonym zasobem.