Przetwarzanie współbieżne. Programowanie równoległe i rozproszone
Laboratorium 7

Cel:

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

Zajęcia:

1. Skopiowanie paczek CzytPis_Pthreads.tgz oraz ProdKons_Pthreads.tgz i ProdKons.tgz, rozpakowanie i uruchomienie programów
2. Utworzenie katalogu roboczego (np. lab_7) i podkatalogu (np. lab_7_pthreads), rozpakowanie paczki CzytPis_Pthreads.tgz i uruchomienie kodu.
3. Przeanalizowanie pseudokodu monitora Czytelnia na slajdach z wykładów oraz struktury kodu w paczce CzytPis_Pthreads.tgz – wykrycie błędu w kodzie poprzez umieszczenie w procedurach pisania i czytania warunku sprawdzającego poprawność aktualnych liczb pisarzy i czytelników.
4. Na podstawie pseudokodu monitora Czytelnia, poprawienie kodu z paczki CzytPis_Pthreads.tgz, tak aby poprawnie rozwiązywać problem czytelników i pisarzy wykorzystując zmienne warunku (funkcja my_read_lock_lock ma odpowiadać funkcji chcę_czytać – czyli protokołowi wejścia do sekcji krytycznej dla czytelnika, my_read_lock_unlock protokołowi wyjścia i podobnie dla pisarza - my_write_lock_lock i my_write_lock_unlock).
5. Przetestowanie działania kodu – w tym poprawności (jak w p. 3). Testowanie, zgodnie z wzorcem w pliku czyt_pis.c, ma polegać na stworzeniu kilku wątków realizujących funkcje czytelnika i pisarza, które w nieskończonej (lub odpowiednio długiej) pętli będą kolejno realizowały swoje funkcje czytania i pisania z prawidłową realizacją wzajemnego wykluczania.
6. Utworzenie podkatalogu (np. lab_7_read_write_locks) i skopiowanie kodu z lab_7_pthreads.
7. Zmodyfikowanie kodu, tak, żeby korzystać z interfejsu zamków do odczytu i zapisu Pthreads (pthread_rwlock_rdlock, pthread_rwlock_wrlock, pthread_rwlock_unlock).
8. Utworzenie podkatalogu roboczego (np. lab_7_java)
9. W podkatalogu, na podstawie pseudokodu monitora Czytelnia, napisanie w Javie klasy Czytelnia pozwalającej na rozwiązanie problemu Czytelników i Pisarzy. W kodzie należy wykorzystać interfejs java.util.concurrent.locks.* i typy Lock() oraz Condition. Należy użyć konstruktorów ReentrantLock() oraz Lock.newCondition() (oraz funkcji lock.hasWaiters(condition) do sprawdzenia czy kolejka uśpionych na danej zmiennej warunku wątków jest pusta).
10. Przetestowanie działania klasy poprzez stworzenie klasy testującej (z funkcją *main*) oraz kilku obiektów klas Czytelnik i Pisarz, które w nieskończonej (lub odpowiednio długiej) pętli będą kolejno realizowały swoje funkcje czytania i pisania. Do stworzenia tych klas można wykorzystać odpowiednio zmodyfikowany kod z paczki ProdKons.tgz

Dodatkowe kroki:

1. Zaimplementować w Javie mechanizm bariery – posługując się uproszonym interfejsem: synchronised, wait(), signal();

Warunki zaliczenia:

1. Obecność na zajęciach i wykonanie co najmniej kroków 1-10
2. Oddanie jednostronicowego sprawozdania z krótkim odręcznym opisem zadania (cel, zrealizowane kroki, wnioski), kodem źródłowym procedury w C i Javie