**Programowanie współbieżne i rozproszone – laboratorium**

**Instrukcja do ćwiczenia laboratoryjnego II**

**Wątki**

Autor: S. Samolej

Rzeszów, 2016

Podziękowania:
Składam podziękowania panu dr inż. Jędrzejowi Ułasiewiczowi za udostepnienie materiałów wykładowych i laboratoryjnych, które były inspiracją do opracowania tych materiałów.

1. Proszę skompilować i uruchomić przykładowy program „thread1.c”. Aby kompilacja tego i pozostałych programów w ćwiczeniu przebiegła pomyślnie trzeba dołączyć w opcjach konsolidatora bibliotekę **pthread** (Build Options->Linker settings->Link Libraries: pthread). W programie pracują 2 wątki (jeden –główny wątek programu, a drugi – nowy jawnie powołany). Proszę zwrócić uwagę technikę powoływania nowych wątków, oczekiwania na zakończenie potomnych wątków oraz na sposób przekazywania parametrów do powoływanych wątków. Wątki współdzielą pamięć procesu je powołującego, więc możliwa jest modyfikacja zmiennych globalnych z poziomu wątków.
2. Proszę skompilować i uruchomić przykładowy program „thread2.c”. Program „udowadnia”, że następuje automatyczne, niewymuszone przełączanie pomiędzy wątkami w czasie ich pracy. Modyfikowana jest współdzielona zmienna i w zależności od jej stanu wątki wypisują pewną informację („1” lub „2”) i zmieniają stan zmiennej. Proszę zwrócić uwagę na fakt ile „cykli” aplikacji mija pomiędzy akcją modyfikacji zmiennej a przełączeniem pomiędzy wątkami.
3. Proszę skompilować i uruchomić przykładowy program „thread3.c”. W programie zastosowano tzw. semafor nienazwany do synchronizacji komunikacji pomiędzy nadawcą a odbiorcą danych. Komunikacja odbywa się przez zmienną dzieloną. Odbiorca oczekuje na zwolnienie semafora, zanim przeprowadzi odczyt ze zmiennej. Proszę zwrócić uwagę na sposób organizacji cyklicznej komunikacji z potwierdzeniem (synchronizacją) operacji modyfikacji zmiennej.
4. Proszę skompilować i uruchomić przykładowy program „mutual_exclus1.c”. Należy zwrócić uwagę na technikę zapewnienia wzajemnego wykluczania do operacji zapisu/odczytu na współdzielonej zmiennej, które odbywają się cyklicznie.
5. Proszę skompilować i uruchomić przykładowy program „thread8a.c”. Program ilustruje metodę „masowego” powoływania wątków.
6. Proszę zaproponować aplikację składającą się z 4 wątków. 3 wątki cyklicznie „produkują” liczby i zapisują je do jednej współdzielonej zmiennej. Czwarty wątek sprawdza cyklicznie stan zmiennej i wypisuje ją na ekranie. Dostęp do współdzielonej zmiennej proszę zrealizować z zastosowaniem wzajemnego wykluczania. Proszę przetestować działanie programu, gdy wątek odbierający będzie szybki (co 1 sek następuje odbiór) a wątki nadające – wolne (co kilka sekund). Proszę przeanalizować działanie aplikacji w sytuacji odwrotnej, gdy wątek czytający jest wolniejszy od nadawców. Proszę zwrócić uwagę, że w tak zorganizowanej aplikacji nie ma żadnej gwarancji na „przechwycenie” wszystkich modyfikacji zmiennych.

7. Do zestawu przykładowych programów został dołączony „prod_cons1.c”. Jest to przykładowe rozwiązanie problemu producenta-konsumenta w oparciu o mechanizm pamięci dzielonej pomiędzy wiele wątków jednego procesu. Proszę o analizę tego rozwiązania.
8. Do zestawu przykładowych programów został dołączony „prod_cons_q1.c”. Aby kompilacja tego programu w ćwiczeniu przebiegła pomyślnie trzeba dołączyć w opcjach konsolidatora bibliotekę **rt** (Build Options->Linker settings->Link Libraries: rt). Program stosuje zestaw funkcji do obsługi kolejek ze standardu POSIX dostępnych w pliku nagłówkowym „mqueue.h”. Proszę się zapoznać z głównymi funkcjami obsługi kolejek i przeanalizować prosty program rozwiązujący problem producenta-konsumenta z zastosowaniem kolejek i procesów.
9. Proszę zaproponować aplikację złożoną z 3 wątków. 2 z nich wysyłają dane do trzeciego przez kolejkę. Trzeci odbiera i identyfikuje (rozpoznaje nadawcę) i wyświetla odebrane komunikaty. Identyfikacja nadawcy może być zrealizowana przez odpowiednie pole w strukturze przesyłanego komunikatu. Proszę zwrócić uwagę, że w tym wypadku otrzymujemy „gotowe” rozwiązanie nie wymagające „ochrony” współdzielonej kolejki z zastosowaniem semaforów. Dostęp do takiej „gotowej” kolejki jest domyślnie realizowany z zasadą wzajemnego wykluczania.
10. Proszę zaproponować aplikację złożoną z 3 wątków. Każdy z nich wykonuje pewne obliczenia i zawiesza swoją pracę do momentu, gdy wszystkie 3 nie dokończą obliczeń. Po zsynchronizowaniu wątków informują one o synchronizacji i kończą pracę.