Narzędzia synchronizacji procesów: semafor, mutex

Problem wzajemnego wykluczania występuje gdy procesy(wątki) rywalizują w dostępie do wspólnych zasobów np. zmiennych globalnych. Można go rozwiązać wykorzystując semafor lub mutex.

ćwiczenie 1:
Zapoznać się działaniem programów zapisanych w plikach źródłowych *p1.c* i *p2.c* podanych w folderze *lekcja3*. Uważnie przeczytać sekcję ćwiczenia zawartą jako komentarz w plikach źródłowych. Wykonać podane ćwiczenia i odpowiedzieć na pytania.

Celem ćwiczenia jest nauczenie się korzystać z narzędzi (semafor, mutex) dostępnych w języku C do synchronizacji wątków w problemie wzajemnego wykluczania.

Zwrócić uwagę:
- Kiedy używamy semafora, a kiedy mutexu
- czym się różni semafor od mutexu
- jak synchronizujemy procesy, w których występują sekcje krytyczne
- jak zdefiniować funkcje bezpieczne ze względu na wielowątkowość

ćwiczenie 2:

W pliku p3.c podano program wielowątkowy, w którym wątki generują liczby losowe. Wątek główny otrzymuje zestaw liczb losowych po jednej liczbie z każdego wątku, a następnie oblicza wartość średnią (arytmetyczną) z tych liczb i wypisuje ją na ekranie. Ogólnie wątki mogą generować w ten sposób wiele zestawów liczb i dla każdego z nich wątek główny powinien wykonać obliczenia. Napisać poprawną synchronizację wątków w tym programie. Rozpatrzyć przypadki:
a) wątki generują jeden zestaw liczb
b) wątki generują wiele zestawów liczb

ćwiczenie 3:

W pliku p4.c podano program wielowątkowy, w którym wątek główny generuje wątki ulotne. Czas działania wątków powinien być losowy (użyć funkcji sleep). Wątek główny powinien kontrolować liczbę działających wątków zdefiniowaną w programie.

Zadania dodatkowe

uwaga:
a) liczba wątków w programach powinna być dowolna ustalona przez #define jeżeli w treści zadania nie określono inaczej.
b) synchronizację programów należy wykonać za pomocą semaforów, mutexów.

1) Problem synchronizacyjny znany jako problem 5 filozofów. Każdy z pięciu filozofów wykonuje dwie czynności: je i myśli. Czynności te są wykonywane w nieskończoność. Filozofowie siedzą przy okrągłym stole. Na stole jest 5 widelców. Każdy filozof może jeść jeżeli posiada dwa widelce. Widelec nie może być w posiadaniu jednocześnie przez więcej niż jednego filozofa. Napisać program który symuluje działanie filozofów. Działanie każdego filozofa wykonuje funkcja odrębnego wątku. Odpowiedni wydruk programu powinien pokazać poprawną synchronizację.

Własność bezpieczeństwa:
- nie dopuścić do zakleszczenia
- nie dopuścić do sytuacji by dwóch filozofów posiadało jednocześnie ten sam widelec

Własność żywotności:
- nie dopuścić do sytuacji w której filozof będzie zagłodzony

sugerowane rozwiązania:
Filozof bierze jeden widelec (gdy jest wolny) a następnie drugi (gdy jest wolny) i zaczyna jeść. Gdy skończy odkłada drugi widelec a następnie pierwszy.
Druga własność bezpieczeństwa będzie zagwarantowana gdy widelec będzie samym mutexem lub chroniony przez mutex.
Czy pierwsza własność bezpieczeństwa (brak zakleszczenia) będzie zapewniona? (znaleźć odpowiedni przeplot)

rozwiązania poprawne:
I) z wykorzystaniem semafora : do stołu dopuścić jednocześnie nie więcej niż 4 filozofów
II) filozofowie o numerach nieparzystych biorą widelce kolejno z prawej i lewej swojej strony, a filozofowie o numerach parzystych biorą widelce w odwrotnej kolejności

Przedstawić obie wersje ((I) i (II)) programu w języku C i proces filozofa w pseudokodzie.