Programowanie współbieżne i rozproszone – laboratorium

Instrukcja do ćwiczenia laboratoryjnego III

Semafory, pamięć dzielona, kolejki komunikatów

Autor: S. Samolej

Rzeszów, 2013

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 „hello_unsync.c". W programie 2 procesy konkurują o standardowe wyjście i w losowych odstępach czasu zapisują kolejne litery fragmentu tekstu. Ponieważ dostęp do zasobu nie jest uregulowany na wyjściu programu otrzymuje się nieczytelny tekst. Istotną rolę w programie odgrywa funkcja **busywait()**, która stosuje funkcję systemową **times()** do odczytu czasu od startu systemu i do blokowania działania procesu.
2. Proszę skompilować i uruchomić przykładowy program „hello_sync.c". Aby kompilacja przebiegła pomyślnie trzeba dołączyć w opcjach konsolidatora bibliotekę **pthread** (Build Options->Linker settings->Link Libraries: pthread). Tym razem zastosowano semafor do synchronizacji pomiędzy procesami. Semafor jest inicjowany wartością 0. Proces macierzysty wywołuje funkcję **sem_wait()** oczekując na zwolnienie semafora. Proces potomny dokonuje zapisu na standardowym wyjściu i wywołuje funcję **sem_post()** zwalniając semafor. Proszę zwrócić uwagę na zastosowane makro **assert()**, które, jeśli program jest skompilowany w trybie DEBUG, to w przypadku wykrycia błędu przerywa działanie programu, wypisuje błąd na standardowym wyjściu błędów.
3. Proszę skompilować programy „shm1.c" i „shm2.c". W obu projektach należy dołączyć plik „shm_com.h" zawierający definicję struktury, która będzie potem rzutowana na współdzielony obszar pamięci. Proszę uruchomić programy komendą konsoli:
   ./shm1 &
   ./shm2
   Programy umożliwiają wysyłanie (shm2) komunikatów do współdzielonego obszaru pamięci i odbieranie ich (shm1). Pamięć dzielona jest zrzutowana na strukturę typu **struct shared_use_st**. Proszę zwrócić uwagę na rolę pola struktury **written_by_you**, które informuje, czy przekazane dane zostały już „skonsumowane".
4. Proszę skompilować programy „msg1.c" i „msg2.c". Proszę uruchomić programy komendą konsoli:
   ./msg2

   ./msg1
   Pomiędzy wywołaniami pierwszego i drugiego programu proszę podać kilka tekstów, które zostaną przekazane do kolejki. Proszę zwrócić uwagę, w jaki sposób seria komunikatów została najpierw zapamiętana w kolejce, a potem prawidłowo odczytana przez odbiorcę.
5. Proszę skompilować i uruchomić program „sysv_msgq.c". Należy zwrócić uwagę na kolejność nadawanych i odbieranych wiadomości.

6. Zaproponować rozszerzenie programów „shm1.c” i „shm2.c” w taki sposób, aby proces zapisu i odczytu do pamięci współdzielonej był sekcją krytyczną chronioną przez semafor.
7. Zaproponować rozszerzenie programów „msg1.c” i „msg2.c” w taki sposób, aby jeden z nich został serwerem-odbiorcą komunikatów generowanych przez kilku producentów-nadawców. Programy mają się komunikować przez kolejkę komunikatów. W programie uwzględnić uprzywilejowanego nadawcę, którego komunikaty będą zawsze od razu odbierane przez program serwera.