Instrukcja laboratoryjna nr 3
Synchronizacja sekcją krytyczną

oprac. Robert Tomaszewski

Program przykładowy ex2.c

```
/* Program demonstrujący użycie obiektu sekcji krytycznej do synchronizacji wątków.
   Obsługa sekcji krytycznej obejmuje następujące kroki/funkcje:
   1) Deklaracja obiektu sekcji
   2) Zainicjowanie obiektu, przydział zasobów - InitializeCriticalSection
   3) Próba przejęcia obiektu, wejścia do sekcji krytycznej (jeśli nieudana to
      wątek zawiesi się do czasu powodzenia operacji - EnterCriticalSection
   4) Opuszczenie sekcji krytycznej (ktoś inny może do niej wejść, przejąć
      obiekt sekcji - LeaveCriticalSection
   5) Zniszczenie obiektu sekcji, uwolnienie zasobów - DeleteCriticalSection
*/

#include <windows.h>
#include <stdio.h>

/* Definicja typu strukturalnego i typu wskaźnikowego na strukturę */
typedef struct
{
   char napis[10];
   int liczba;
} TPARAM, *PTPARAM;

/* Deklaracja obiektu sekcji krytycznej - można stworzyć kilka obiektów */
CRITICAL_SECTION sekcja;

/* Funkcja wątku */
DWORD WINAPI proc(LPVOID arg)
{
 int i,licznik1=0; /* prywatne dane wątku */
 PTPARAM parametr; /* zmienna na parametr przekazany do wątku - u nas struktura */

 /* Próba wejścia do sekcji krytycznej */
 EnterCriticalSection(&sekcja);
 for (i=0;i<10;i++)
 {
  ++licznik1;
  printf("WATEK w sekcji krytycznej: licznik watku: %d\n",licznik1);
 }
 parametr = (PTPARAM) arg; /* przypisanie argumentu do zmiennej prywatnej */
 printf("Parametr watku:%s, %u\n",parametr->napis,parametr->liczba++);
 /* Opuszczenie obiektu sekcji krytycznej */
 LeaveCriticalSection(&sekcja);
}

int main(void)
{
 TPARAM zm; /* tą zmienną przekażemy do funkcji wątków */
 DWORD id;
 int licznik=0;

 /* Wypełnienie pól struktury */
 printf("Wpisz jakis tekst (do 10 znakow):");
 scanf("%s",zm.napis);
 printf("A teraz liczbe calkowita dodatnia:");
 scanf("%u",&zm.liczba);
 /* Zainicjowanie obiektu sekcji krytycznej */
 InitializeCriticalSection(&sekcja);
 /* Utworzenie wątku - bez zapamiętania uchwytu */
 if (CreateThread(NULL,0,proc,&zm,0,&id) != NULL)
  printf("Watek utworzony!\n");
 /* Program/wątek główny też coś robi */
 for (licznik=0;licznik<=5;licznik++)
  printf("PROGRAM: Licznik globalny programu: %d\n",licznik);
```

```
  printf("Program czeka na wejscie do sekcji krytycznej...\n");
 /* Wątek główny czeka za zakończenie wątku potomnego - zwolnienie sekcji krytycznej */
 EnterCriticalSection(&sekcja);
 printf("Parametr watku (zmieniony przez watek):%s, %u\n",zm.napis,zm.liczba);
 LeaveCriticalSection(&sekcja);
 DeleteCriticalSection(&sekcja);
 system("PAUSE");
 return 0;
}
```

Program przykładowy ex3.c

```
/* Poniższy program został zaczerpnięty z witryny WWW Microsoftu (MSDN) i demonstruje
   nieprawidłowe wykorzystanie obiektów sekcji krytycznej - następuje w nim zakleszczenie
*/
#include <windows.h>
#include <stdio.h>

CRITICAL_SECTION cs1,cs2; /* deklaracja dwóch obiektów sekcji krytycznej */

/* Prototyp funkcji */
DWORD WINAPI ThreadFn(LPVOID);

int main(void)
{
  DWORD iThreadID;
  InitializeCriticalSection(&cs1);
  InitializeCriticalSection(&cs2);
  /* Zwróć uwagę na poniższy, oryginalny sposób utworzenia wątku - nie będziemy
     potrzebować uchwytu, więc od razu go zwalniamy - TO NIE ZABIJA WĄTKU! Powoduje
     jedynie, że nie możemy skorzystać z funkcji wątkowych wymagających podania HANDLE */
  CloseHandle(CreateThread(NULL,0,(LPTHREAD_START_ROUTINE)ThreadFn,NULL,0,&iThreadID));
  while(TRUE){
   EnterCriticalSection(&cs1);
   printf("\nWatek1 jest w sekcji 1, ale nie w 2.");
   EnterCriticalSection(&cs2); /* moment potencjalnego zakleszczenia! */
   printf("\nWatek1 jest w sekcji 1 i 2!");
   LeaveCriticalSection(&cs2);
   printf("\nWatek1 opuscil sekcje 2, ale jest w sekcji 1.");
   LeaveCriticalSection(&cs1);
   printf("\nWatek1 opuscil obie sekcje krytyczne...");
   Sleep(20);
  };
  return(0);
}

/* Funkcja wątku - będzie rywalizować z wątkiem głównym o obiekty sekcji krytycznej */
DWORD WINAPI ThreadFn(LPVOID lParam)
{
  while(TRUE)
  {EnterCriticalSection(&cs2);
   printf("\nWatek2 jest w sekcji 2, ale nie w 1.");
   EnterCriticalSection(&cs1); /* moment potencjalnego zakleszczenia! */
   printf("\nWatek2 jest w sekcji 2 i 1!");
   LeaveCriticalSection(&cs1);
   printf("\nWatek2 opuscil sekcje 1, ale jest w sekcji 2.");
   LeaveCriticalSection(&cs2);
   printf("\nWatek2 opuscil obie sekcje krytyczne...");
   Sleep(20);
  };
}
```

ZADANIE: Zapoznaj się z przykładowym programem ex2.c demonstrującym sposób przekazywania parametrów do funkcji wątków oraz zasadę synchronizacji poprzez sekcje krytyczne. Napisz program, który pobiera od użytkownika 20 liczb (zapisując je w zmiennej tablicowej), tworzy dwa wątki i przesyła do nich (jako parametry ich funkcji) po połowie tablicy. Wątki sortują otrzymane połówki tablicy, informują program/wątek główny o zakończeniu swojej pracy, zaś ten wyświetla posortowane połówki, następnie scala je w jedną posortowaną całość i ją wyświetla. Do synchronizacji wątków roboczych z wątkiem głównym użyj

obiektów sekcji krytycznych.

Wskazówka:

W przypadku, gdyby wątek główny zyskiwał dostęp do sekcji krytycznych przed wątkami roboczymi można uśpić (funkcją Sleep, tuż przed EnterCriticalSection) na krótko wątek główny, aby upewnić się, że planista procesów w systemie operacyjnym umożliwi wątkom wejście do sekcji krytycznych przed wątkiem głównym.

PRZYDATNE ŹRÓDŁA:

Opis wątków i procesów w Windows (interesują nas tylko wątki - threads):
http://msdn2.microsoft.com/en-us/library/ms684841.aspx
Pisanie programów wielowątkowych w Windows:
http://msdn2.microsoft.com/en-us/library/y6h8hye8(VS.71).aspx
Synchronizacja wielobieżności w programach Windows (interesują nas sekcje krytyczne, semafory i zdarzenia – critical section, semaphores, events)
http://msdn2.microsoft.com/en-us/library/ms686353.aspx
Opisy biblioteczne funkcji i struktur używanych w programowaniu wielowątkowym:
http://msdn2.microsoft.com/en-us/library/aa908719.aspx