Instrukcja laboratoryjna nr 2
Synchronizacja zmiennymi globalnymi

oprac. Robert Tomaszewski

Program przykładowy ex.c

```
/* Przykład - prosty program wielowątkowy demonstrujący zasadę tworzenia wątków
   (tutaj 2 wątki potomne) przez program (wątek!) główny; zwróć uwagę na wartości
   zmiennych licznikowych w trakcie działania programu:
   licznik - zmienna globalna, używana przez wątek główny
   licznik1 - zmienna statyczna w funkcji wątków, współdzielona przez wątki
   licznik2 - zmienna automatyczna w funkcji wątków, prywatna dla każdego wątku
   W przykładzie brak jakiejkolwiek synchronizacji i komunikacji między wątkami.
   Brak również przekazywania parametrów do funkcji wątków.
*/

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

int licznik=0;

/* Funkcja wątku(ów) */
DWORD WINAPI proc(LPVOID arg)
{
 static int licznik1=0;
 int i,licznik2=0;

 for (i=0;i<10;i++)
 {
  /* poniższa pętla inkrementuje oba liczniki, wyświetla ich wartości oraz identyfikator
     aktualnie wykonywanego wątku */
  ++licznik1; ++licznik2;
  printf("WATEK %u: licznik statyczny watku: %d\n",GetCurrentThreadId(),licznik1);
  printf("WATEK %u: licznik prywatny watku: %d\n",GetCurrentThreadId(),licznik2);
 }
}

int main(void)
{
 HANDLE watek1 = NULL, watek2 = NULL;
 DWORD nr; /* można wykorzystać pojedynczą zmienną identyfikującą wątek */

 printf("Proba utworzenia 2ch watkow...\n");
 watek1 = CreateThread(NULL,0,proc,NULL,0,&nr);
 if (watek1 != NULL) /* tak można sprawdzić powodzenie operacji tworzenia wątku */
  printf("Watek o numerze %u utworzony\n",nr);
 watek2 = CreateThread(NULL,0,proc,NULL,0,&nr);
 if (watek2 != NULL)
  printf("Watek o numerze %u utworzony\n",nr);
 for (licznik=0;licznik<=5;licznik++)
  printf("WATEK GLOWNY: Licznik globalny programu: %d\n",licznik);
 Sleep(2000); /* sztuczne opóźnienie - niezalecane w wątku głównym! */
 system("PAUSE");
 CloseHandle(watek1); /* zwalniamy zasoby (uchwyty) po wątkach */
 CloseHandle(watek2);
 return 0;
}
```

ZADANIE: Zapoznaj się z powyższym przykładowym programem ex.c demonstrującym sposób tworzenia wątków. Przepisz/skopiuj powyższy program do środowiska programistycznego, skompiluj i uruchom. Zwróć uwagę na wyświetlane informacje o wartości poszczególnych zmiennych licznikowych – dają one wyobrażenie o przeplotach wykonania wątku głównego z wątkami potomnymi. Zauważ, że przy niektórych wykonaniach programu niektóre wątki wykonują kilka swoich instrukcji pod rząd (bez oczekiwanego, „idealnego" wzajemnego przeplatania się; da się to rozpoznać po wyświetlanych identyfikatorach wątków).

Program przykładowy ex1.c

```
/* Program wielowątkowy demonstrujący prostą synchronizację między wątkami
   opartą na wspólnej zmiennej globalnej (stoper). Wada rozwiązania w rozważanym
   przykładzie - wątek który jako pierwszy zakończy pętlę FOR, ustawi zmienną
   "stoper" i w ten sposób może uniemożliwić wykonanie do końca drugiego wątku
   poprzez odblokowanie wątku głównego (wątek główny może się w ten sposób
   zakończyć terminując swoich potomków).
   Dodatkowo przykład demonstruje sposób przekazywania parametrów do funkcji
   wątków - tutaj pewnej struktury (dwa pola: ciąg znaków oraz liczba).
*/

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

/* Definicja typu strukturalnego i typu wskaźnikowego na strukturę,
   wykorzystamy te definicje w funkcji wątków do przekazania parametrów
*/

typedef struct
{
   char napis[10];
   int liczba;
} TPARAM, *PTPARAM;

/* Zmienne globalne, w tym "stoper" służący do synchronizacji */
int licznik=0, stoper=0;

/* Funkcja wątku - kilka wątków może wykonywać tą samą lub różne funkcje */
DWORD WINAPI proc(LPVOID arg)
{
 static int licznik1=0; /* tą zmienną wątki będą współdzielić */
 int i,licznik2=0; /* a to są już prywatne dane wątków */
 PTPARAM parametr; /* zmienna na parametr przekazany do wątku - u nas struktura */

 for (i=0;i<10;i++)
 {
  ++licznik1; ++licznik2;
  printf("WATEK %u: licznik statyczny watku: %d\n",GetCurrentThreadId(),licznik1);
  printf("WATEK %u: licznik prywatny watku: %d\n",GetCurrentThreadId(),licznik2);
 }
 /* pierwszy z wątków który zakończy pętlę ustawi stoper */
 parametr = (PTPARAM) arg; /* przypisanie argumentu do zmiennej prywatnej */
 /* Wypisanie pola napisowego oraz liczby - liczbę po wszystkim inkrementujemy */
 printf("Parametr:%s, %u\n",parametr->napis,parametr->liczba++);
 stoper=1; /* ustawienie zmiennej synchronizującej */
}

int main(void)
{
 HANDLE watek1, watek2; /* uchwyty dla wątków */
 DWORD nr; /* zmienna na numer ID wątku */
 TPARAM zm; /* tą zmienną przekażemy do funkcji wątków */

 /* Wypełnienie pól struktury */
 printf("Wpisz jakis tekst (do 10 znakow):");
 scanf("%s",zm.napis);
 printf("A teraz liczbe calkowita dodatnia:");
 scanf("%u",&zm.liczba);
 /* Próba utworzenia wątków - można nie zapamiętywać uchwytów, jeśli nie będą
    nam potrzebne (nie wywołamy żadnej funkcji wątkowej, która oczekuje podania uchwytu
    do identyfikacji wątku
 */
 watek1 = CreateThread(NULL,0,proc,&zm,0,&nr);
 if (watek1 != NULL)
  printf("Watek o numerze %u utworzony\n",nr);
 watek2 = CreateThread(NULL,0,proc,&zm,0,&nr);
 if (watek2 != NULL)
```

```
  printf("Watek o numerze %u utworzony\n",nr);
 /* Program główny też coś robi */
 for (licznik=0;licznik<=5;licznik++)
  printf("PROGRAM: Licznik globalny programu: %d\n",licznik);
 /* Program główny zatrzymuje się - aktywne oczekiwanie, czyli prosta pętla */
 while (stoper != 1)
  printf("Program czeka...\n");
 CloseHandle(watek1);
 CloseHandle(watek2);
 system("PAUSE");
 return 0;
}
```

ZADANIE: Zapoznaj się z przykładowym programem ex1.c demonstrującym sposób przekazywania parametrów do funkcji wątków oraz zasadę synchronizacji poprzez zmienne globalne. 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. Zastanów się jak poinformować program/wątek główny o zakończeniu pracy przez wszystkie wątki w sytuacji, kiedy wykonują tą samą funkcję.

Wskazówka - jedno z możliwych rozwiązań:

Zmienna synchronizacyjna może być wyzerowaną tablicą o liczbie pozycji równej liczbie utworzonych wątków. Każdy wątek po zakończeniu pracy ustawia (0 -> 1) jedną pozycję tablicy (zrób użytek ze zmiennych prywatnych i wspólnych w funkcji wątku do indeksacji tablicy). Aktywne oczekiwanie programu/wątku głównego kończy się w momencie ustawienia wszystkich pozycji.

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