Совместное использование данных несколькими процессами через DLL

Предоставление каждому процессу собственного адресного пространства порождает проблему совместного использования (говорят – разделения) одних данных несколькими процессами.

Очень эффективный прием заключается в создании отдельной секции данных в DLL и установке для этой секции атрибута "**shared**", после чего эта секция будет совместно использоваться всеми загрузившими DLL процессами.

По умолчанию компилятор помещает инициализированные перемененные в секцию с именем "**.data**", а неинициализированные – в секцию "**.bss**". По умолчанию им обоим присваиваются атрибуты **R**ead и **W**rite, разрешающие и чтение, и запись, но не исполнение или совместное использование! Изменить разрешения можно с помощью компоновщика, запустив его следующим образом:

link *Имя obj-файла* ***/SECTION:имя секции,атрибуты***.

Пример. Создадим DLL, экспортирующую некоторую переменную, например, так:

_declspec(dllexport) char buff[666] = "Hello, Sailor\n"

и откомпилируем ее:

"*cl* ***mydll.c***",

"*link* ***mydll.obj*** */DLL /SECTION*:*.data,RWS*"

Теперь создадим два приложения – пусть одно из них что-то пишет в переменную *g_count*, а другое – читает это оттуда. Итак:

Листинг 1 – Демонстрация использования совместно используемой памяти через разделяемую (shared) секцию в DLL

```
// Test_w.c
// Приложение-писатель
#include <string.h>
#include <windows.h>
// Подключаем DLL неявной компоновкой.
// Можно и явной, но это будет длиннее
__declspec(dllimport) char buff[100];
```

```
main()
{
  strcpy(&buff[0],"Hello, World!\n");   // Пишем
  Sleep(6000);         // Пауза 6 сек
}

// Test_r.c
// Приложение-читатель. Совсем простое :-)
#include <stdio.h>
__declspec(dllimport) char buff[100];
main(){printf("%s\n",&buff[0]);}
```

Откомпилируем обе программы с подключением библиотеки, а затем запустим приложение-писатель и следом за ним, пока последнее не успело завершиться, приложение-читатель. На экране появится 10. Это работает! И все бы было хорошо, да вот возникает одна проблема – целиком "зашаривать" весь сегмент данных неудобно, да и небезопасно –вдруг некорректно работающая программа, "дорвавшись" до адресного пространства нашей DLL "вырубит весь лес под чистую"! Гораздо удобнее представлять совместный доступ только к некоторым, специально на то предназначенным переменным.

Чтобы не трогать секцию ".data" создадим свою собственную! Достаточно лишь включить в исходный текст программы директиву

*#pragma data_seg("**Имя секции**")*,

например, так:

Листинг 2 – Демонстрация создания собственной секции в DLL

```
// mysection.c
// buff_data помещается в секцию .data по умолчанию
_declspec(dllexport)
   char buff_data[333]="Hello, DATA!\n";

// Создаем новую секцию - .ASHARE
// Точка перед именем не обязательна - просто так
// принято
#pragma data_seg(".ASHARE")

// Теперь переменная buff_shared принудительно
// помещается в секцию .SHARED
_declspec(dllexport)
   char buff_shared[777]="Hello, a_SHARED!\n";
```

```
// Указываем компилятору вновь помещать переменные
// в секцию по умолчанию, т.е. .data
#pragma data_seg()

// buff_data2 помещается в секцию .data вслед
// за buff_data
_declspec(dllexport)
   char buff_data2[555]="Hello, DATA2\n";
```

Откомпилируем пример "*cl **mysection.c***", "*link **mysection.obj** /DLL /SECTION:.ASHARE,RSW*" и с помощью dumpbin убедимся, что новая секция успешно создана и содержит в себе единственную переменную buff_shared, а все остальные находятся в секции ".data". Это можно сделать так: "*dumpbin **mysection.dll** /ALL*".

Теперь модифицируем приложение-читатель и приложение-писатель так, чтобы они работали с переменной buff_shared. Для этого достаточно заменить "buff" на "buff_ shared". Работа с импортируемыми элементами не зависит от того, какая секция их экспортирует!

Разделяемые переменные должны быть инициализированы. В противном случае компилятор Microsoft Visual C++ (да и другие тоже))поместит все неинициализированные переменные в секцию "**.bss**". Рассмотрим это на следующем примере:

```
#pragma data_seg(".ASHARE")
_declspec(dllexport) int test;
#pragma data_seg()
```

Компилятор, невзирая на все предписания, поместит переменную test в секцию ".bss"! Выход из этой ситуации – инициализация "test" некоторым значением.

Замечание. Секции с атрибутами совместного использования доступны всем процессам, загрузившим эту DLL, и Ваши данные не защищены от посягательств со стороны некорректно работающих приложений.