**Лабораторная работа №9**

**Задание.**

Задание. Построить DLL библиотеку Windows, функции которой позволяют запомнить внутри нее текст данных, а затем - в другой функции - прочитать этот текст в вызывающую программу. Использовать эту библиотеку в двух параллельно работающий процессах, где основная программа использует динамическую компоновку времени загрузки, один из программных модулей – либо DLL библиотека, либо основная программа - написан на языке ассемблера, а другой из этих модулей – на языке Си; библиотека DLL обеспечивает выдачу сообщений о подключении к ней процесса и об отключении его от нее.

**Ход работы:**

На рисунке 1 представлен вызов первого процесса с обращением к DLL библиотеке

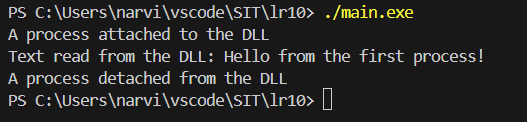
****

Рисунок 1

На рисунке 2 представлен вызов второго процесса с обращением к DLL библиотеке

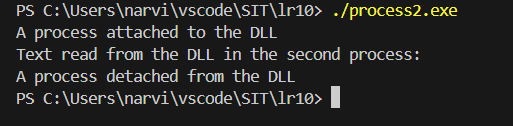


Рисунок 2

**Пояснение работы:**

1. **Запоминание текста**:  
   Функция save\_text сохраняет текст в статическом массиве stored\_text внутри DLL. Этот массив доступен из всех процессов, использующих DLL.
2. **Чтение текста**:  
   Функция read\_text возвращает указатель на сохраненный текст.
3. **Логирование подключения и отключения**:  
   При загрузке или выгрузке DLL логируются события DLL\_PROCESS\_ATTACH и DLL\_PROCESS\_DETACH.

Таким образом, два процесса могут взаимодействовать через общую DLL.

**Приложение**

**main.c**

#include <windows.h>

#include <stdio.h>

// Определяем функции, которые будем использовать из DLL

typedef void (\*save\_text\_func)(const char\*);

typedef const char\* (\*read\_text\_func)();

int main() {

    // Загрузка DLL

    HINSTANCE hDll = LoadLibrary("textdll.dll");

    if (!hDll) {

        printf("Failed to load the DLL\n");

        return 1;

    }

    // Получение указателей на функции

    save\_text\_func save\_text = (save\_text\_func)GetProcAddress(hDll, "save\_text");

    read\_text\_func read\_text = (read\_text\_func)GetProcAddress(hDll, "read\_text");

    if (!save\_text || !read\_text) {

        printf("Failed to get function addresses\n");

        FreeLibrary(hDll);

        return 1;

    }

    // Сохранение текста в DLL

    save\_text("Hello from the first process!");

    // Чтение текста из DLL

    const char\* text = read\_text();

    printf("Text read from the DLL: %s\n", text);

    // Освобождение DLL

    FreeLibrary(hDll);

    return 0;

}

**process2.c**

#include <windows.h>

#include <stdio.h>

typedef const char\* (\*read\_text\_func)();

int main() {

    // Загрузка DLL

    HINSTANCE hDll = LoadLibrary("textdll.dll");

    if (!hDll) {

        printf("Failed to load the DLL\n");

        return 1;

    }

    // Получение указателя на функцию чтения

    read\_text\_func read\_text = (read\_text\_func)GetProcAddress(hDll, "read\_text");

    if (!read\_text) {

        printf("Failed to get the function address\n");

        FreeLibrary(hDll);

        return 1;

    }

    // Чтение текста из DLL

    const char\* text = read\_text();

    printf("Text read from the DLL in the second process: %s\n", text);

    // Освобождение DLL

    FreeLibrary(hDll);

    return 0;

}

**textdll.c**

#include <windows.h>

#include <stdio.h>

#define EXPORT \_\_declspec(dllexport)

// Хранилище текста, доступное между вызовами

static char stored\_text[1024] = {0};

// Функция для записи текста

EXPORT void save\_text(const char\* text) {

    strncpy(stored\_text, text, sizeof(stored\_text) - 1);

    stored\_text[sizeof(stored\_text) - 1] = '\0';  // Гарантируем нуль-терминатор

}

// Функция для чтения текста

EXPORT const char\* read\_text() {

    return stored\_text;

}

// Логирование подключения процесса

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul\_reason\_for\_call, LPVOID lpReserved) {

    switch (ul\_reason\_for\_call) {

        case DLL\_PROCESS\_ATTACH:

            printf("A process attached to the DLL\n"); // Сообщение при подключении

            break;

        case DLL\_PROCESS\_DETACH:

            printf("A process detached from the DLL\n"); // Сообщение при отключении

            break;

    }

    return TRUE;

}