Заголовочный файл для библиотеки

```
// Пример заголовочного файла для библиотеки
// Файл MyLib.h
// define MYDLL_EXPORTS for export

#ifdef MYLIB_EXPORTS
#define MYLIB_API __declspec(dllexport)
#else
#define MYLIB_API __declspec(dllimport)
#endif
// Экпортируемый из MyLib.dll класс
class MYLIB_API CMyLib {
public:
        CMyLib(void);
        // TODO: add your methods here.
};
extern MYLIB_API int nMyLib;
MYLIB_API int fnMyLib(void);
```

Файл исходного текста библиотеки(1)

```
// MYDLL.cpp : Defines the entry point for the DLL application.
#define MYDLL_EXPORTS
#include <windows.h>
#include "MYDLL.h"
BOOL APIENTRY DllMain( HANDLE hModule,  DWORD  ul_reason_for_call,
                       LPVOID lpReserved )
{
    switch (ul_reason_for_call)
                {
                        case DLL_PROCESS_ATTACH:

                        case DLL_THREAD_ATTACH:
                        case DLL_THREAD_DETACH:
                        case DLL_PROCESS_DETACH:
                                break;
    }
    return TRUE;
}
```

Файл исходного текста библиотеки(2)

```
// This is an example of an exported variable
MYDLL_API int nMYDLL=24;

// This is an example of an exported function.
MYDLL_API int fnMYDLL(void)
{
        return 49;
}

// This is the constructor of a class that has been exported.
// see MYDLL.h for the class definition
CMYDLL::CMYDLL()
{
        return;
}
```

DEF-файл проекта библиотеки

```
LIBRARY   MYDLL
DESCRIPTION "My Dll Creation Demo"
EXPORTS
   fnMYDLL  @1
   nMYDLL @2
```

Клиент с явной компоновкой(1)

```
// MYDLL_ExplicitUse.cpp.
#include <windows.h>
……….
……….
……….
case WM_COMMAND:
wmId    = LOWORD(wParam);
wmEvent = HIWORD(wParam);
// Parse the menu selections:
switch (wmId)
{
```

Клиент с явной компоновкой(2)

```
case IDM_FILE_CALL:
{
typedef UINT (* LPFN_1_TYPE)(VOID);
HINSTANCE hDll;                 // Handle to DLL
LPFN_1_TYPE lpfnDllFunc1; // Function pointer
hDll=LoadLibrary("MYDLL");
```

Клиент с явной компоновкой(3)

```
if (hDll != NULL)
  {
  lpfnDllFunc1 = (LPFN_1_TYPE)
                          GetProcAddress(hDll,"fnMYDLL");
  if (!lpfnDllFunc1)
     { // handle the error
     FreeLibrary(hDll);
     return ERROR_FUNCTION_NOT_FOUND;
     }
   else
     {// call the function
     nFromLib = lpfnDllFunc1();
     FreeLibrary(hDll);
     }
  }
```

Клиент с явной компоновкой(4)

```
HDC hdc=GetDC(hWnd);
//nFromLib=fnMYDLL();

sprintf(szBuff,"Call to fnMYDLL() from Explicitly Linked Dll:
%d",
                    nFromLib);
TextOut(hdc,100,100,szBuff,lstrlen(szBuff));
ReleaseDC(hWnd,hdc);
}
break;
}
break;
```

Клиент с неявной компоновкой

```
// MYDLL_ImplicitUse.cpp

#include <windows.h>

#include "MYDLL.h"
```

Клиент с неявным подключением библиотеки

```
case IDM_FILE_CALL:

TCHAR szBuff[200];

int nFromLib=0;

nFromLib=fnMYDLL();
HDC hdc=GetDC(hWnd);
 wsprintf ( szBuff,
            "Call to fnMYDLL() from Implicitly Linked Dll: %d",
            nFromLib );


TextOut(hdc,100,100, szBuff, lstrlen(szBuff) );
ReleaseDC(hWnd,hdc);
```

СОЗДАНИЕ СЕРВЕРА (DLL)

СОЗДАНИЕ СЕРВЕРА (DLL)

1) Заголовочный файл с экспортируемыми прототипами, структурами и идентификаторами (символьными именами)

2) Исходные файлы C/C++ в которых реализованы функции и определены переменные

3) Компилятор создает OBJ-файл из каждого исходного файла C/C++

4) Компоновщик собирает **DLL** из OBJ-модулей

5) Если DLL экспортирует хотя бы одну переменную или функцию, компоновщик создает и **LIB - файл**.

СОЗДАНИЕ КЛИЕНТА (EXE)

СОЗДАНИЕ КЛИЕНТА (EXE)

6) Заголовочный файл с импортируемыми прототипами структурами и идентификаторами

7) Исходные файлы *C/C++*, из которых вызываются импортируемые функции и переменные

8) Компилятор создает OBJ-файл из каждого исходного файла *C/C++*.

9) Используя OBJ модули и LIB-файл и учитывая ссылки на импортируемые идентификаторы компоновщик собирает **EXE-модуль** (в котором также размещается таблица импорта — список необходимых DLL и импортируемых идентификаторов).

СОЗДАНИЕ СЕРВЕРА (DLL) и КЛИЕНТА (EXE)