Gustaf Nilstadius

1 File1_Funkce1 is present in the file1.obj, but the linker wants the name "?File1_Funkce1@YAHHH@Z" witch is the C++ name of the function. Thereby the function is not found.

2 The functions to export is listed in the obj file and ban be viewed with dumpbin. The linker uses the information to link.

3 In the library_static the functions from the obj files are present and declared in the coff symbol table. The dynamic library does not since the functions are loaded on demand and we only need to have knowledge of their presence.

4 The imports section contains the information about imports ("/IMPORTS" in dumpbin util). The imported symbols are "File1_Funkce1".

5 The library must be in the PATH (environment variable) or in the working directory.

6 *See bottom of document for code*

7 I do not get the error, even tho i have checked with dumpbin that library.lib has one more function than library.dll

8

9
```
  00000006: 6A 02              push        2
  00000008: 6A 01              push        1
  0000000A: E8 00 00 00 00     call        _File1_Funkce1
* 0000000F: 83 C4 08           add         esp,8
  00000012: 89 45 FC           mov         dword ptr [ebp-4],eax
  00000015: 6A 02              push        2
  00000017: 6A 01              push        1
  00000019: E8 00 00 00 00     call        _File1_Funkce2@8
  0000001E: 89 45 F8           mov         dword ptr [ebp-8],eax
  00000021: 8B 45 FC           mov         eax,dword ptr [ebp-4]
```

At line * the esp is added by 8, this is not performed after the second function. That is because the File1_Funkce2 follows the _stdcall calling convention where the function cleans up the stack.

10 The Funkce2 cleans up the stack by itself, this is done at "ret   8". A positive ret will add to the stack. That gives the same result as the marked line in question 9. A function with _cdelc will not clear the stack, this is donne by the caller.

file4.cpp

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

int main(int argc, char** argv)
{
        HMODULE hModule = NULL;
        int (*pfnFile1_Funkce1)(int, int) = NULL;
        int (__stdcall *pfnFile1_Funkce2)(int,int) = NULL;
        int (*pfnFile2_Funkce1)(int, int) = NULL;
        int (__stdcall *pfnFile2_Funkce2)(int,int) = NULL;

        hModule = LoadLibrary( TEXT("library.dll") );
        if( hModule )
        {
                pfnFile1_Funkce1 = (int (*)(int,int))GetProcAddress( hModule, "File1_Funkce1" );

                if( pfnFile1_Funkce1 )
                {
                        printf("Soucet: %d.\n", pfnFile1_Funkce1(1,2) );
                }
                else
                {
                        printf("File1_Funkce1: Nenalezena. Chyba %d.\n", GetLastError());
                }

                pfnFile1_Funkce2 = (int (__stdcall*)(int,int))GetProcAddress( hModule, "_File1_Funkce2@8" );

                if( pfnFile1_Funkce2 )
                {
                        printf("Soucet: %d.\n", pfnFile1_Funkce2(1,2) );
                }
                else
                {
                        printf("File1_Funkce2: Nenalezena. Chyba %d.\n", GetLastError());
                }

                //FILE2
                pfnFile2_Funkce1 = (int (*)(int,int))GetProcAddress( hModule, "?File2_Funkce1@@YAHHH@Z" );

                if( pfnFile2_Funkce1 )
                {
                        printf("Soucet: %d.\n", pfnFile2_Funkce1(1,2) );
                }
                else
                {
                        printf("File2_Funkce1: Nenalezena. Chyba %d.\n", GetLastError());
                }

                pfnFile2_Funkce2 = (int (__stdcall*)(int,int))GetProcAddress( hModule, "?File2_Funkce2@@YGHHH@Z" );

                if( pfnFile2_Funkce2 )
                {
                        printf("Soucet: %d.\n", pfnFile2_Funkce2(1,2) );
                }
                else
                {
                        printf("File2_Funkce2: Nenalezena. Chyba %d.\n", GetLastError());
                }

                FreeLibrary( hModule );
                hModule = NULL;
        }

        return 0;
}
```