# DLLs (dynamic loaded libraries) mit MingW erstellen

Autor: Michel D. Schmid

Datum: April 2, 2009

## Contents

**1 Einführung** **1**
1.1 Open-Source Tools . . . . . 1

**2 Beispiel 1: Hello World!** **1**
2.1 Der Source-Code . . . . . 1
2.2 Das Übersetzen des Codes . . . . . 2

**3 Beispiel 2: weitere Funktionen** **3**
3.1 Der Source-Code . . . . . 3
3.2 Das Übersetzen des Codes . . . . . 5

**4 Beispiel 3: Zugriff auf Klassen** **5**

**5 Literaturverzeichnis** **5**

# 1 Einführung

Das Erstellen von dynamisch geladenen Bibliotheken für die Windows-Plattform mit Open-Source Tools ist nach wie vor kein einfaches Unterfangen. Aus meiner Sicht liegt das an den spärlichen Informationen, die es dazu bisher auf dem Internet gibt.

## 1.1 Open-Source Tools

Mit folgenden Tools sind die hier aufgeführten Anweisungen getestet worden:

1. MinGW-5.1.3 (**Min**imal **G**nu for **W**indows), darin enthalten sind:
   (a) GCC 3.4.5, mehr wird bisher für das Beispiel nicht benötigt!

# 2 Beispiel 1: Hello World!

## 2.1 Der Source-Code

Auch hier gibts zuerst das "'Hello World'" Beispiel. Dazu benötigen wir drei verschiedene Dateien:

1. example_dll.h
2. example_dll.cpp
3. hello.cpp, enthält die main-Funktion

Die Datei example_dll.h sieht dabei wie folgt aus:

```
00001 #ifdef BUILD_DLL
00002 /* DLL export */
00003 #define EXPORT __declspec(dllexport)
00004 #else
00005 /* EXE import */
00006 #define EXPORT __declspec(dllimport)
00007 #endif
00008
00009 extern "C"
00010 {
00011     EXPORT void hello(void);
00012 }
```

Die Datei example_dll.cpp hat folgenden Inhalt:

```
00001 #include <stdio.h>
00002 #include "example_dll.h"
00003
00004 EXPORT void hello(void) {
00005     printf ("Hello\n");
00006 }
```

Und zuletzt der Inahlt von hello.cpp:

```
00001 extern "C"
00002 #include <windows.h>
00003 #include <stdio.h>
00004 int main () {
00005     /*Typedef the hello function*/
00006     typedef int(*pfunc)(void);
00007 //            typedef int(*PFNAPI2)(void);
00008 // PFNAPI2 hello;
00009
00010     /*Windows handle*/
00011     HINSTANCE hdll;
00012
00013     /*A pointer to a function*/
00014     pfunc hello;
00015
00016     /*LoadLibrary*/
00017     hdll = LoadLibrary("message.dll");
00018
00019     /*GetProcAddress*/
00020     hello = (pfunc)GetProcAddress(hdll, "hello");
00021
00022     /*Call the function*/
00023     hello();
00024     return 0;
00025
00026 }
```

## 2.2 Das Übersetzen des Codes

So, wer nun den Source-Code und einen GNU-C++-Compiler auf dem PC hat (kopiert oder erstellt!) sollte in der Lage sein, dass Beispiel wie unten angegeben zu kompilieren.

Zuerst wird der Objekt-Code der erwünschten DLL erstellt. Dazu muss auf der Kommando-Zeile folgender Befehl eingegeben werden:

```
g++ -c -DBUILD_DLL example_dll.cpp
```

Anschliessend wird nun die eigentliche DLL erstellt. Hier mit dem Namen "message.dll".

```
g++ -shared -o message.dll example_dll.o -Wl,–out-implib,message.a
```

Auf diese Eingabe hin erscheint eine Meldung mit dem Inhalt: "Creating library file: message.a"

Die Compiler-Option **-DBUILD_DLL** wird beim GNU-Compiler dazu eingesetzt, dass die Funktionen innerhalbe der DLL als *dllexport* deklariert werden. Durch diesen Schritt werden also die Funktionen aus der DLL exportiert und anderen Applikationen zugänglich gemacht.
Die Option **-shared** hingegen wird dazu benutzt, dass der Compiler eine .dll erstelt anstelle einer .exe.
Beinahe am Ende der Kommando-Zeile erscheint noch eine Linker-Option: **–out-implib**. Diese wird verwendet, um eine Bibliothek zu erstellen, die später importiert werden kann. Wird jedoch lediglich benötigt, falls nicht alle Teile mit demselben Compiler übersetzt wurden.

Als nächstes folgt nun die Kommando-Zeile, um die .exe zu erstellen.

```
g++ -o hello.exe hello.cpp
```

Somit sollte nun eine ausführbare Datei mit dem Namen *hello.exe* erstellt worden sein. Wer den Code von *hello.exe* schon studiert hat, dem fällt auf, dass wir in solchen Situationen nicht um die Windows-API herum kommen, zumindest nicht um einen kleinen Teil dieser.

# 3 Beispiel 2: weitere Funktionen

## 3.1 Der Source-Code

Für dieses Beispiel bleiben wir bei den drei Dateien aus Beispiel eins und erweitern diese ein Wenig:

1. example_dll.h
2. example_dll.cpp
3. hello.cpp, enthält die main-Funktion

Die Datei example_dll.h sieht nun wie folgt aus:

```
00001 #ifdef BUILD_DLL
00002 /* DLL export */
00003 #define EXPORT __declspec(dllexport)
00004 #else
00005 /* EXE import */
00006 #define EXPORT __declspec(dllimport)
00007 #endif
00008
00009 extern "C"
00010 {
```

```
00011     EXPORT void hello(void);
00012     EXPORT void printString(inti);
00013     EXPORT int square(inti);
00014 }
```

Wir exportieren also zwei weitere Funktionen. Diese müssen natürlich entsprechend implementiert werden:

```
00001 #include <iostream>
00002 #include <stdio.h>
00003 #include "example_dll.h"
00004
00005 EXPORT void hello(void) {
00006     printf ("Hello\n");
00007 }
00008
00009 EXPORT void printString(inti) {
00010     std::cout « i « "\n";
00011 }
00012
00013 EXPORT int square(inti) {
00014     return i * i ;
00015 }
```

Und zuletzt nochmals der angepasste Inhalt von hello.cpp:

```
00001 extern "C"
00002 #include <windows.h>
00003 #include <stdio.h>
00004 #include <iostream>
00005 int main () {
00006     /*Typedef the hello function*/
00007     typedef void(*pfunc)(void);
00008     typedef void(*pfunc2)(int);
00009     typedef int(*pfunc3)(int);
00010
00011     /*Windows handle*/
00012     HINSTANCE hdll;
00013
00014     /*A pointer to a function*/
00015     pfunc hello;
00016     pfunc printString;
00017     pfunc square;
00018
00019     /*LoadLibrary*/
00020     hdll = LoadLibrary("message.dll");
00021
00022     /*GetProcAddress*/
00023     hello = (pfunc)GetProcAddress(hdll, "hello");
00024     /*Call the function*/
00025     hello();
00026
00027     /*GetProcAddress*/
00028     printString = (pfunc2)GetProcAddress(hdll, "printString");
00029     /*Call the function*/
```

```
00030     printString(100);
00031
00032     /*GetProcAddress*/
00033     square = (pfunc3)GetProcAddress(hdll, "square");
00034     /*Call the function*/
00035     intiSquare = square(10);
00036     std::cout « "iSquare:  " « iSquare « "\n";
00037
00038
00039     return 0;
00040
00041 }
```

Weitere Kommentare werde ich zum oben aufgeführten Code nicht geben. Es sind lediglich minimale Änderungen, die verstanden werden sollten.

### 3.2 Das Übersetzen des Codes

Erfolgt wie im vorhergehenden Beispiel.

## 4 Beispiel 3: Zugriff auf Klassen

Folgt in Kürze (was auch immer dies bei einem Familienvater bedeutet:-)

## 5 Literatur erzeichnis

Folgende Links wurden bisher verwendet:

[1] http://sig9.com/node/35
[2] http://www.users.fh-sbg.ac.at/.../.../20erstellen.pdf