Windows

Zunächst einmal sei darauf hingewiesen das sich dieses Tutorial auf die Bildung von Boost unter Windows beschränkt. Es wird eine Version für Visual Studio .NET 2005, auch Version 8.0 genannt, gebildet. Wenn Sie die Boost Bibliotheken für Linux bilden möchten, finden Sie am Ende dieser Seite einen Link der Ihnen eventuell weiterhilft. Desweiteren existiert in Visual Studio .NET 2005 ein praktikabler und bequemer Weg die Boost Bibliotheken in sein eigenes Projekt einzubinden. Dazu stellt die IDE eine neue Menü-Option namens *'Project from Existing Code'* zur Verfügung.

In einer Ausgabe von MSDN TV wird eine Video-Anleitung zur Einbindung von Boost unter Visual Studio .NET 2005 vorgestellt. Am Ende dieses Tutorials finden Sie den entsprechenden Download-Link zu dem Video.

Als fortgeschrittener C++ Programmierer werden Sie vielleicht schon mal von www.Boost.org gehört haben oder ihnen wurden schon mal die Boost Bibliotheken empfohlen. Ursprünglich vom C++ Standardisierungskomitee gegründet ist Boost zu einer großen Sammlung von freien C++-Bibliotheken herangewachsen, die ein breites Spektrum an portablen Problemlösungen bieten. Boost versucht die herausragenden Eigenschaften der generischen Programmierung und der Metaprogrammierung aufzunehmen und zu erweitern. Die Boost Bibliotheken sind vollständig plattformunabhängig und bilden heute die Grundlage für viele auf C++ basierende Projekte. Zahlreiche Funktionen aus Boost wurden im Laufe der Zeit in den offiziellen Standard aufgenommen und damit fester Bestandteil der C++-Programmiersprache.

Sie sehen, die Boost Bibliotheken bieten ihnen als C++ Programmierer ein breites Spektrum an Möglichkeiten für ihre Projekte. Bevor Sie die C++ Boost Bibliotheken nun erstellen können, benötigen Sie einige Pakete.

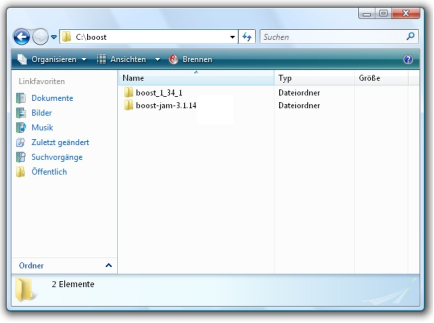
- Laden Sie sich unter http://sourceforge.net/projects/boost/files/boost/ die aktuelle Boost Distribution herunter. Sie haben die Wahl zwischen verschiedenen Paketen. In unserem Fall werden wir das .zip Paket benutzen.

- Nun benötigen Sie noch den Build-Manager mit dem Namen Bjam. Sie können diesen über die Kommandozeile selbst kompilieren. Alternativ dazu finden Sie unter http://sourceforge.net/projects/boost/files/boost-jam/ eine kompilierte Version. Seit der Version 1.43.0 von Boost, ist Bjam im Paket von Boost enthalten. Ein separater Download von Bjam ist damit nicht mehr notwendig.

Nachdem Sie nun alle notwendigen Pakete zum Bau von Boost besitzen sollten Sie einen Ordner erstellen. In unserem Beispiel werden wir den Ordner "boost" auf dem Laufwerk C: erstellen. Unser Pfad lautet demnach C:boost.

Nun kopieren Sie das Paket boost-jam-3.1.14.zip in unseren zuvor erstellten Ordner hinein. Extrahieren Sie nun auch die Boost Distribution und kopieren Sie diese ebenfalls in unseren Ordner. Stellen Sie sicher das Sie genügend freien Speicherplatz auf der Festplatte zur Verfügung haben. Ein vollständiger Bau von Boost nimmt ungefähr 3 GB Speicherplatz in Anspruch.

Nun müsste, je nachdem welche Version Sie geladen haben, Ihr Verzeichnis wie folgt aussehen:



BJam kompilieren

Um den BJam Manager selbst zu kompilieren, müssen wir die entsprechende Batch-Datei aufrufen. Bevor Sie mit dem Bau von BJam beginnen, müssen Sie sicherstellen das Ihre Umgebung entsprechend korrekt konfiguriert ist. Gehen Sie dazu in den Visual Studio Ordner - in meinem Fall ist das der Pfad C:ProgrammeMicrosoft Visual Studio 8VCbin - und führen Sie Stapelverarbeitungsdatei vcvars32.bat aus. Bedenken Sie das Sie unter Windows Vista über entsprechende Administratorrechte verfügen müssen.

Das Ausführen von vcvars32.bat bewirkt das alle Umgebungsvariablen und andere Optionen für den Compiler richtig konfiguriert werden. Dieser Schritt ist besonders dann notwendig, wenn Sie verschiedene C++-Compiler auf Ihrem System installiert haben.

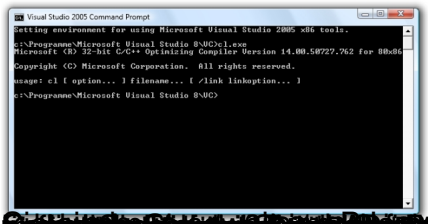
Nun sollten Sie überprüfen ob alle Variablen richtig gesetzt wurden. Öffnen Sie wieder die Kommandozeile, indem Sie z.B. in Ihrem Programmordner die Visual Studio Mappe suchen, anschließend den Ordner Visual Studio Tools öffnen und Visual Studio 2005 Command Prompt ausführen. Anschließend tippen Sie cl oder cl.exe ein. Der Microsoft C++-Compiler müsste nun fehlerfrei aufgerufen werden. Ist das nicht der Fall müssen Sie Ihre Visual Studio 2005 Installation noch einmal überprüfen. Besonders unter Windows Vista verläuft die Installation von Visual Studio 2005 oft nicht sauber, so dass es hier zu Fehlern kommen kann. Diese Probleme wird Microsoft mit Erscheinen von Visual Studio 2008 beheben.

Sollte es auch bei Ihnen Probleme geben müssen Sie die Umgebungsvariablen manuell überprüfen und setzen. Öffnen Sie dazu die erweiterten Systemeinstellungen unter Ihrem Arbeitsplatz und klicken Sie auf Umgebungsvariablen. Sehen Sie nach ob die Variable VS80COMNTOOLS vorhanden ist und auf den richtigen Installationspfad zeigt. In unserem Fall befindet sich die Installation der Tools unter dem folgenden Pfad: C:Program FilesMicrosoft Visual Studio 8Common7Tools

Falls die Variable unter den Systemvariablen nicht vorhanden ist, legen Sie sie an. Darüberhinaus müssen Sie die PATH Variable überprüfen. Bearbeiten Sie die Variable und fügen die folgenden Pfade hinzu: C:Program FilesMicrosoft Visual Studio 8Common7IDE;C:Program FilesMicrosoft Visual Studio 8VCbin

Oftmals wird beim Ausführen von cl.exe der fatale Fehler angezeigt das die mspdb80.dll nicht gefunden werden konnte. Die oben ausgeführten Schritte beheben das Problem. Installieren Sie auf keinen Fall die DLL in Ihrem System32 Ordner. Die Bibliothek befindet sich in Ihrem Visual Studio Verzeichnis und wird bei korrekter Konfiguration auch richtig ausgeführt. Manchmal müssen Sie neben den bereits genannten Pfaden noch den Pfad zu den C++ Headern explizit angeben. In meinem Fall ist das der folgende Pfad: C:Program FilesMicrosoft Visual Studio 8VCinclude

Nachdem Sie die Schritte ausgeführt haben können Sie die Dateien vcvars32.bat und vcvarsall.bat noch einmal ausführen. Letztere befindet sich eine Verzeichnisebene über der ersten Datei. Rufen Sie nun den C++-Compiler noch einmal mit cl auf. Folgendes sollte nun auf Ihrer Konsole zu sehen sein.



.bat

Die Boost Bibliotheken bauen

Sie haben nun alle mehr oder weniger aufwendigen Schritte hinter sich gebracht und Ihre Umgebung ist bereit um die Boost Bibliotheken zu kompilieren. Über die Kommandozeile rufen wir nun die bjam.exe auf und teilen dem Programm mit das es Bibliotheken sowohl für den Debug als auch für den Release Modus bilden soll. Zusätzlich geben wir die Optionen threading und runtime-link dynamic an. Auf diese Weise wird ein Debug/Release Build mit Multithreading und dynamisch/statischen Libs für Visual Studio 2005 (8.0) gebildet.

```
C:boost>bjam "-sBUILD=debug release static/dynamic multi" "-sTOOLS=vc-8_0"
```

Sie können den Build auch jeweils einschränken:

```
C:boost>bjam "-sBUILD=release dynamic multi" "-sTOOLS=vc-8_0"
```

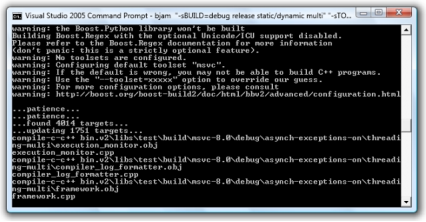
Alternativ zu den oben genannten Kommandozeilen Befehlen können sie auch zwei Batch-Dateien mit folgendem Inhalt erstellen, in das Boost Stammverzeichnis kopieren und ausführen.

```
boostdebugbuild.bat: bjam "-sBUILD=debug dynamic multi" "-sTOOLS=vc-8_0"
boostreleasebuild.bat: bjam "-sBUILD=release dynamic multi" "-sTOOLS=vc-8_0"
```

Die Variable sTOOLS kann auch mit anderen Daten gefüttert werden, so dass Sie bei Bedarf auch Bibliotheken für Visual Studio 2003.NET (7.1) erstellen können. Spezifizieren Sie dazu einfach die genaue Version der Entwicklungsumgebung.

Die Boost Bibliotheken werden gebildet

Haben Sie alles richtig gemacht, werden nun die Boost Libraries gebildet. Da die Boost Quellcodes massiven Gebrauch von Templates machen, benötigen die Compiler oft ziemlich lange für die Kompilierung. Unter Umständen kann dieser Prozess 120 Minuten und länger dauern. Je nachdem über welche Systemresourcen Sie verfügen und welche Bibliotheken gebildet werden sollen. Als Richtwert können Sie mit 180 Minuten auf einem Pentium 4 mit 2,53 GHz und 1024 MB Ram rechnen.



Verwendung der Boost Bibliotheken

Die fertigen Lib's und DLL's finden Sie unter boost_1_34_1bin.v2. Das Einbinden in Ihre C++ Entwicklungsumgebung realisieren Sie entsprechend über die Projekteigenschaften. Je nachdem mit welchen Libraries sie Linken möchten und welche Run-Time Routinen (/MD, /ML, /MT, /LD (Use Run-Time Library)
) sie verwenden, müssen sie die entsprechenden Optionen setzen. Boost bietet die Möglichkeit über Präprozessor Direktiven das dynamische Linken explizit anzugeben. Nutzen sie dazu die folgenden Direktiven:

[code xml:lang="cpp"]#define BOOST_ALL_DYN_LINK #define BOOST_LIB_DIAGNOSTIC[/code]

Alternativ dazu können sie vor Aufruf der Boost Header eine Header-Datei in den Precompiled Headers includieren. Eine entsprechende Header-Datei ist diesem Tutorial angehängt.

Vergessen sie nicht die entsprechenden DLL's in das Verzeichnis ihres Programmes zu kopieren wenn sie die Option "Multi-threaded DLL (/MD)" verwenden. Achten sie außerdem darauf das sie nicht statische Versionen mit dynamsichen Versionen der Run-Time Libraries vermischen oder sie erhalten einen Compilerfehler. Die Header Dateien müssen wie gewohnt ebenfalls in Ihr Projekt includiert werden.

Bei statischen Run-Time Linking "Multi-threaded Debug (/MT)" wird der Code entsprechend statisch ihrem Programm hinzugefügt, so dass sie eine seperate DLL nicht benötigen.

Herzlichen Glückwunsch! Ihnen stehen nun die umfangreichen Funktionen von C++ Boost zur Verfügung. Abschließend möchten wir diese testen. Dazu nutzen wir das boost filesystem mit diesem kleinen Programm.

[code xml:lang="cpp"]// Boost Dateien includieren #include "boost/filesystem/operations.hpp" #include "boost/filesystem/fstream.hpp" #include using namespace std; // Benutze fs für den Boost filesystem Namensraum namespace fs = boost::filesystem; int main() { fs::remove_all("foobar"); fs::create_directory("foobar"); ofstream

file("foobar/cheeze.txt"); file