

Was gehört (nicht) ins SVN?

- Keine Dateien die aus anderen Dateien generiert werden (z.B. beim der Kompilierung)
 - .suo, .ncb (Intellisense), .user, das komplette Debug und Release Verzeichnis
- Nur Dateien die vom Kompilieren nötig sind
 - .cpp .h .vcproj .sln

Präprozessor, Linker, Compiler Bibliotheken

Compiler

1. Präprozessor
Textersetzungen gemäß Präprozessoranweisungen
2. Übersetzer [Compiler]
Übersetzt Programmfragmente in C++ zu Programmfragmenten in Binärcode
3. Verknüpfer [Linker]
Verknüpft Binärprogrammteile zu einem Programm

Präprozessor

- Macht Textersetzungen *vor* dem Übersetzen
- alle Anweisungen für den Präprozessor fangen mit # an
 - `#include "file.h"` fügt die Header-Datei file.h ein
 - `#define DEBUGOUT 1` ersetzt überall im Programmtext DEBUGOUT mit 1
 - `#ifdef DEBUGOUT`
Code1
#else
Code2
#endif
fügt im Programmtext Code1 ein falls DEBUGOUT definiert wurde, ansonsten Code2

Präprozessor

```
int m_myInt;  
...
```

class1.h

```
#include "class1.h"  
...
```

class2.h

```
#include "class1.h"  
#include "class2.h"  
...
```

class3.h

- `m_myInt` wird zweimal definiert
- Durch Präprozessoranweisungen können wir das verhindern

```
#ifndef CLASS1_H  
#define CLASS1_H  
int m_myInt;  
...  
#endif
```

class1.h

Präprozessor

- Textersetzung abhängig von Präprozessordefinitionen

```
#ifdef DEBUGOUT
    std::cout << debugMessage << std::endl;
#else
    std::cout << normalMessage << std::endl;
#endif
```

class1.cpp

```
#ifdef _WIN32
#pragma warning ( disable : 4244 )
#endif
```

class2.cpp

Bibliotheken [Libraries]

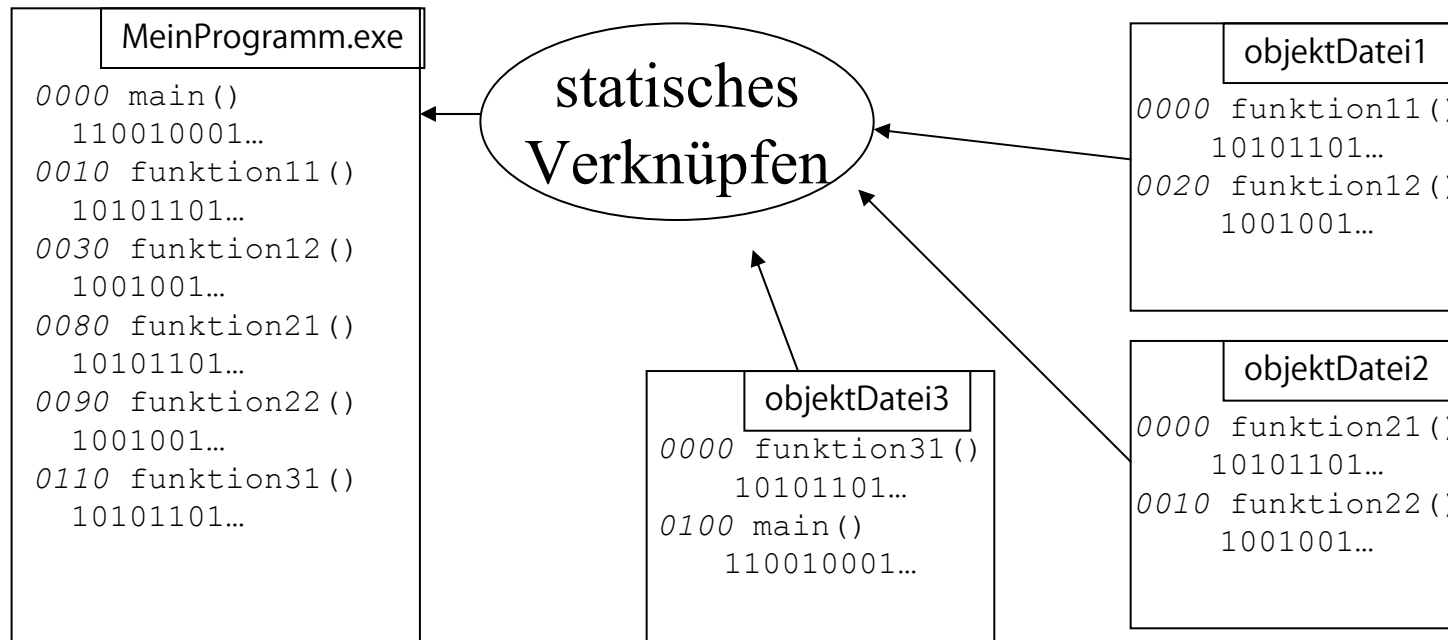
- Bibliotheken (*.lib) sind vorkompilierte Binärdateien ohne Einstiegspunkt [Entrypoint]
- Zeitvorteil, weil Programmteile, die sehr oft benutzt werden, nicht immer wieder neu kompiliert werden müssen.
- Rechtlich einwandfrei, weil Programmteile weitergegeben werden ohne den Programmtext zeigen zu müssen.
- Programmiertechnischer Vorteil, weil weitergegebene Programmteile nicht geändert werden können.

Einstiegspunkt?!

- Der Einstiegspunkt [Entrypoint] eines Programms ist der Teil, an dem das Programm starten soll. Unter C++ ist es die globale Funktion `int main()`, die es selbstverständlich nur einmal geben darf
- Ein Programm ohne Einstiegspunkt ist allein nicht lauffähig

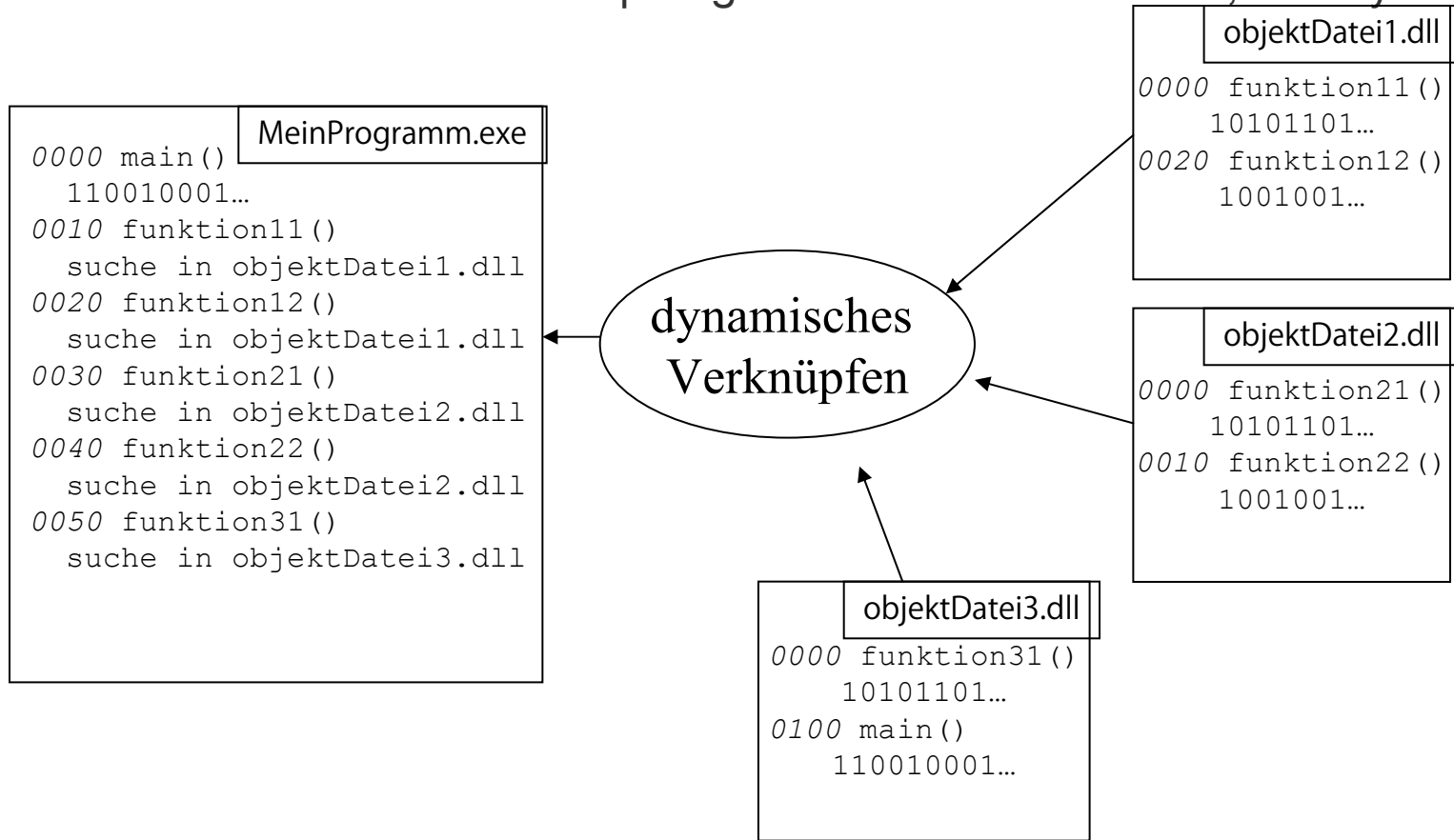
Verknüpfen (link)

- Das eigentliche Programmerstellen übernimmt der Linker
- Er kopiert den Code aus den Bibliotheken und ersetzt im Code die symbolischen Funktionen mit konkreten Adressen



Verknüpfen (link)

- Die zweite Art der Verknüpfung funktioniert zur Laufzeit, d.h. dynamisch



Gegenüberstellung

statisches Verknüpfen

- einfachste (auch älteste) Methode
- ein große Datei
- neue Version der Bibliothek → ganze Datei muss neu generiert werden
- keine Probleme mit verschiedenen Versionen

dynamisches Verknüpfen

- komplizierter Mechanismus, der auf unterschiedlichen Betriebssystemen unterschiedlich angesprochen wird
- Weniger Speicherbedarf
- neue Version der Bibliothek → nur Bibliothek muss neu generiert werden
- Realisierung von Plugins möglich

Tipps zu Compiler-/Linkerfehlern

- meist wurde in den angegebenen Objektdateien ein Symbol (Funktion, Variable, Klasse, ...) nicht gefunden
- Der Grund dafür ist meistens
 - ein Schreibfehler
 - falscher oder fehlender Namensraum
 - `virtual / static` vergessen
- Der Compiler prüft ***nie*** nach, ob eine deklarierte Funktion (im Header xx.h) tatsächlich (in der Implementierung xx.cpp) implementiert wurde
- Der Compiler beschwert sich aber ***immer***, wenn eine nicht deklarierte Funktion implementiert wird
- Manchmal hilft es, ***alles*** (inklusive Bibliothek) komplett neu zu bauen

TODO-Liste für fremde Bibliotheken

- im Programmtext den Header einbinden, in dem die gewünschten Objekte oder Funktionen bereitgestellt werden
- dem Linker den/die Dateinamen der Bibliotheken mitteilen, damit diese mit dem Programm verknüpft werden können
- den Pfad des Headers **und** der Bibliothek angeben

(Für uns) wichtige Bibliotheken

- C++ Standardbibliothek
- Boost
- Qt – grafische Buntzeroberfläche
- OpenGL – 3D Grafik (eigentlich nur eine API Spezifikation)
- OpenCV – Computer Vision und Bildverarbeitung
- ITK – Segmentierung und Registrierung
- Xerces - XML

C++ Standardbibliothek (std::)

- Fester Bestandteil von C++
- Container, Strings, Datenströme, Mathematische Funktionen
- Viele Funktionen verwenden Templates
- Maßgeblich beeinflusst von HPs Standard Template Library (STL). Oft auch synonym mit STL verwendet.
- <http://www.cppreference.com>

Boost

- Enthält viele allgemein verwendbare Teilbibliotheken z.B. `smart_ptr`, `filesystem`, `graph`, `thread`
- Alle Bibliotheken unterliegen einem Review-Prozess
- Einige Teilbibliotheken werden zukünftig Bestandteil der C++ Standardbibliothek werden (TR1, TR2)
- <http://www.boost.org>

std::vector

- Dynamische Arrays (normale Arrays haben eine feste Größe)

array.cpp

```
int arraySize;
double myDoubleArray[arraySize];
...
for (int i=0; i<arraySize; i++) {
    myDoubleArray[i] += 5.0;
}
```

vector.cpp

```
std::vector<double> myDoubleVector;
...
for (std::vector<double>::iterator iter = myDoubleVector.begin();
     iter != myDoubleVector.end(); iter++ ) {
    *iter += 5.0;
}
```

std::map

- Höhenbalancierter Suchbaum

```
std::map<std::string, int> telefonbuch;  
telefonbuch["Ryu Hilla"] = 28917058;  
std::cout << telefonbuch["Ryu Hilla"] << std::endl;
```

map.cpp

weiteres

- `Std::list`
- `Std::queue`
- `Std::priority_queue`
- `fstream, iostream`
- Sortieren, Permutieren, Union und Difference, Komplexe Zahlen

boost::shared_ptr

- Wenn mehrerer Objekte Pointer auf das selbe Objekt verwenden ist es manchmal schwer zu entscheiden, wann das Objekt gelöscht werden darf
- Smart Pointer schaffen Abhilfe
- Boost::shared_ptr ist eine Implementierung von Smart Pointern
- Das Object wird gelöscht, sobald kein Pointer mehr auf das Objekt existiert

```
myObject* objPtr = new myObject();
```

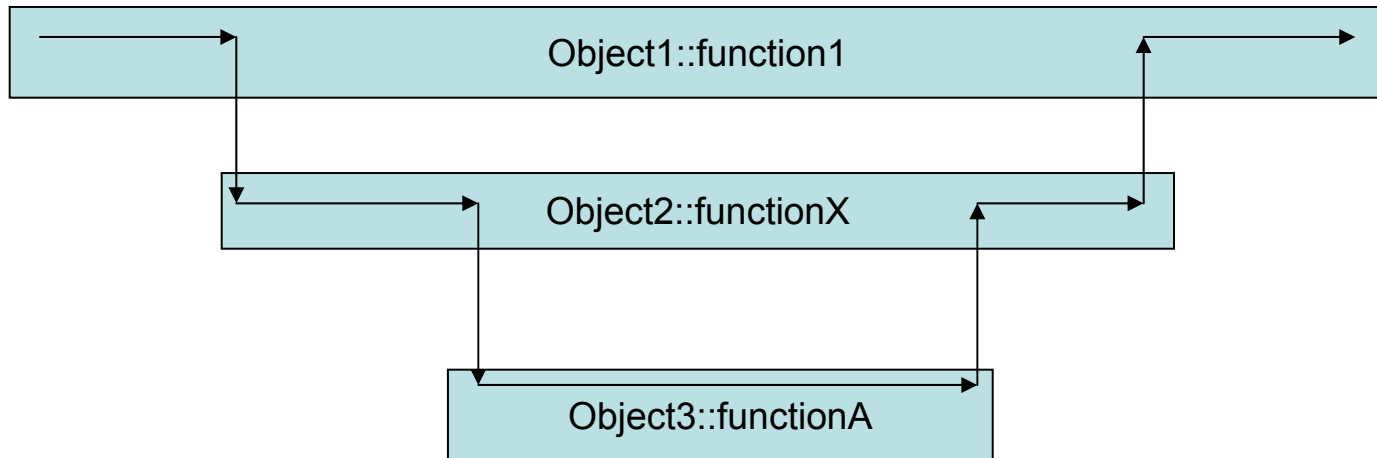
Pointer.cpp

```
boost::shared_ptr<myObject> objSharedPtr(new myObject());
```

SharedPtr.cpp

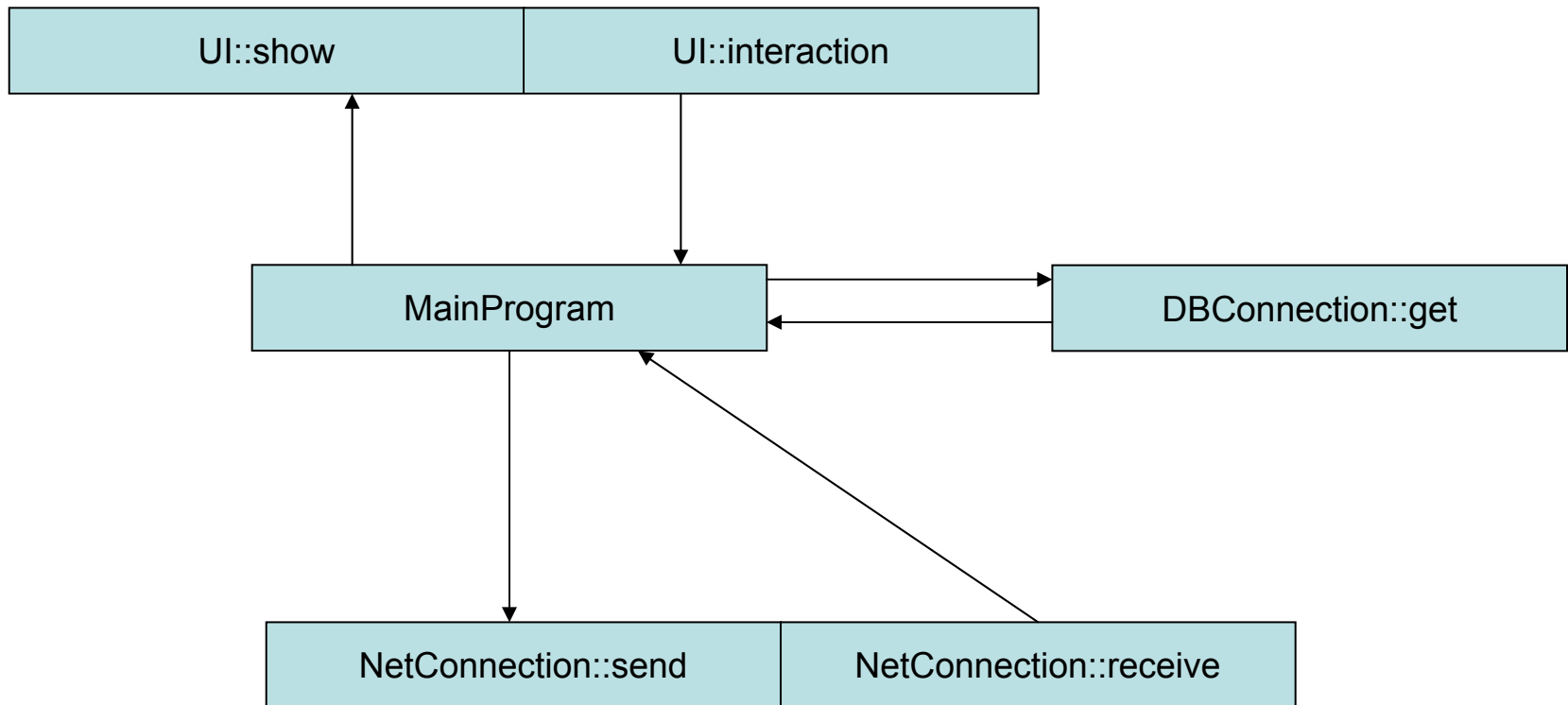
boost::signal

- Einfacher Programmfluss



boost::signal

- Komplexerer Programmfluss



boost::signal

- Signals and Slots
- Ereignisgesteuerter Programmfluss
- Geprägt durch Qt
- Eine Signal kann mit mehreren Slots verbunden werden
- Wird ein Signal ausgesandt, werden alle dazugehörigen Slots aufgerufen