technische universität
dortmund

Software-Engineering für langlebige Systeme

Thomas Ruhroth | Dortmund
SoSe 2015

PA

- HA 5
-

DLLs und SOs – Möglichkeiten der Anbindung

- Statisches Binden
  - Angabe beim Compilieren
  - Wird häufig direkt in das Executable eingebaut
  - Unter Windows nur mit Object Files möglich (.o)
- Dynamisches Binden einer beim Compilieren bekannten DLL
  - Angabe beim Compiler
  - Für Linux siehe Vorlesungsfolien
  - Windows: Ein paar Klicks in der GUI
- Dynamisches Binden einer beim Compilieren *nicht* bekannten DLL
  - Laden und binden als Operationen im Code
  - Schwierig, aber zum Glück selten gebraucht

Windows Klicken in der GUI – Aus MS-Hilfe

- Um die in der DLL (Dynamic Link Library) erstellten mathematischen Routinen zu verwenden, müssen Sie auf die DLL verweisen. Wählen Sie dazu im Menü Projekt die Option Verweise... aus. Erweitern Sie im Dialogfeld Eigenschaftenseiten den Knoten Allgemeine Eigenschaften, und wählen Sie Verweise aus. Wählen Sie dann die Schaltfläche Neuen Verweis hinzufügen... aus. Weitere Informationen zum Dialogfeld Verweise... finden Sie unter Verweise, Allgemeine Eigenschaften, Dialogfeld "<Projektname>-Eigenschaftenseiten".
- Das Dialogfeld Verweis hinzufügen wird angezeigt. Dieses Dialogfeld listet alle Bibliotheken auf, auf die Sie verweisen können. Auf der Registerkarte Projekt werden alle Projekte in der aktuellen Projektmappe und sämtliche darin enthaltenen Bibliotheken aufgelistet. Wählen Sie auf der Registerkarte Projekte das Projekt MathFuncsDll aus. Klicken Sie auf OK. Weitere Informationen zum Dialogfeld Verweis hinzufügen finden Sie unter Dialogfeld "Verweis hinzufügen".
- Um auf die Headerdateien der DLL zu verweisen, müssen Sie den Includeverzeichnispfad ändern. Erweitern Sie dazu im Dialogfeld Eigenschaftenseiten den Knoten Konfigurationseigenschaften, dann den Knoten C/C++, und wählen Sie dann Allgemein aus. Geben Sie neben Zusätzliche Includeverzeichnisse den Pfad des Speicherorts der Headerdatei MathFuncsDll.h ein.
- DLLs werden von der ausführbaren Datei erst zur Laufzeit geladen. Sie müssen dem System mitteilen, wo MathFuncsDll.dll zu finden ist. Verwenden Sie dafür die PATH-Umgebungsvariablen. Erweitern Sie dazu im Dialogfeld Eigenschaftenseiten den Knoten Konfigurationseigenschaften, und wählen Sie Debuggen aus. Geben Sie neben Umgebung Folgendes ein: PATH=<path to MathFuncsDll.dll file>, wobei <path to MathFuncsDll.dll file> durch den tatsächlichen Speicherort von MathFuncsDll.dll ersetzt wird. Drücken Sie auf OK, um alle vorgenommenen Änderungen zu speichern.

Die Hausaufgabe - Teile

- Main – Klasse
- Java-Wrapper
- C-Wrapper

Java-Wrapper

```
package lsys;

public final class ReactorWrapper {

    static{
        System.loadLibrary("reactorwrapper");
    }

    public native double temp();
public native void selectTemp(double temp);
public native double getSelectedTemp();
public native double getTemp();
public native void stop();
public native void start();
}
```

C-Wrapper

```
#include "lsys_ReactorWrapper.h"
#include "reactor.h"

JNIEXPORT jdouble JNICALL Java_lsys_ReactorWrapper_temp
  (JNIEnv * env, jobject obj)
{
return getTemp();
};
```

```
/*
 * Class:     lsys_ReactorWrapper
 * Method:    selectTemp
 * Signature: (D)V
 */
JNIEXPORT void JNICALL Java_lsys_ReactorWrapper_selectTemp
  (JNIEnv * env, jobject obj, jdouble d)
{
selectTemp(d);
};
```

```
/*
 * Class:     lsys_ReactorWrapper
 * Method:    getSelectedTemp
 * Signature: ()D
 */
JNIEXPORT jdouble JNICALL Java_lsys_ReactorWrapper_getSelectedTemp
  (JNIEnv * env, jobject obj){
return getSelectedTemp();
};
```

```
/*
 * Class:     lsys_ReactorWrapper
 * Method:    getTemp
 * Signature: ()D
 */
JNIEXPORT jdouble JNICALL Java_lsys_ReactorWrapper_getTemp
  (JNIEnv * env, jobject obj){
return getTemp();
};
```

```
/*
 * Class:     lsys_ReactorWrapper
 * Method:    stop
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_lsys_ReactorWrapper_stop
  (JNIEnv * env, jobject obj){
stop();
};
```

```
/*
 * Class:     lsys_ReactorWrapper
 * Method:    start
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_lsys_ReactorWrapper_start
  (JNIEnv * env, jobject obj){
start();
};
```

Main

```
public static void main(String[] args) {


System.out.println(System.getProperty("java.library.path"));
int i = 0;
ReactorWrapper reactor = new ReactorWrapper();
reactor.selectTemp(85.0f);
reactor.start();
    for(i = 0;i < 200; i++) {
        System.out.println("Temp: " + reactor.getTemp() + " Grad Celcius");
    }
reactor.stop();
    for(i = 0;i < 200; i++) {
        System.out.println("Temp: " + reactor.getTemp() + " Grad Celcius");
    }
}
```

Entscheidungstabelle

- Zusammen an der Tafel

- Punkte:
  - Auswahl der Alternativen,
  - die Auswahl der Kriterien,
  - Analyse der Kriterien,
  - die Dokumentation,
  - die Definitionen und
  - die Auswertung der Tabelle

Prüfungssimulation

- A – Prüfling
  - Fragen beantworten
- B – Prüfer
  - Fragen
    - Wichtig: Fragen sind Einstiegsfragen.
      Immer mindestens eine Nachfrage!
    - Nachfragen, evtl. Tips geben
- C – Protokollant
  - Fragen (insbesondere Nachfragen) kurz notieren
  - Kurz Antwort festhalten und „bewerten“ (Richtig/Falsch)

A

- Welche Arten von Updates gibt es?
- <Eine Art genauer erklären lassen>
- Was ist eine Graphtransformation? Aus welchen Teilen besteht Sie?
- Was ist das Reengineering-Hufeisen?

B

- Welche Planungsfehler haben wir in der Vorlesung behandelt?
- Was ist eine Co-Evolution?
- Gibt es Regeln bei der Auswahl von Alternativen in einem Entscheidungsprozess?
- Was sind die Herausforderungen beim Dekompilieren von
  - Java
  - C?

C

- Was ist der templatebasierte Ansatz bei Transformationen?
- Gibt es Regeln für die Entscheidungsqualität in einem Entscheidungsprozess?
- Welche Probleme sind bei der Durchführung eines Updates zu beachten?
- Was sind die Vorteile einer Schichtenarchitektur?

Frageduell

- Zwei Teams
- Antwortzeit 2 Min
- Wenn Antwort nicht gewusst, dann 2 Min zum erklären
  - Sonst Punkt für die Gegner
- Thomas ist Schiedsrichter