Modulare Programmierung und Bibliotheken

Proseminar-Vortrag

am 24.06.2011 von

Ludwig Eisenblätter

Modulare Programmierung und Bibliotheken

Inhaltsübersicht

Motivation

Warum modulare Programmierung?

- Hohe Komplexität und großer Umfang von Softwareprojekten
- Erweiterung der imperativen Programmierung
- Aufteilung eines Projekts in logische Teilblöcke
- Wiederbenutzung von Code
- Nutzung von Bibliotheken

Modulare Programmierung - Allgemeines

Modularisierung von Software

Obere Abstraktionsebene
(Programmierung im Großen)

- Gestaltung und Anordnung von Modulen
- Definition und Zusammenführung von Programmeinheiten
- Interaktion von Modulen

Untere Abstraktionsebene
(Programmierung im Kleinen)

- Steueranweisungen
- Funktionale Abstraktion
- Datenabstraktion (Datenstrukturen)

Modulare Programmierung - Allgemeines

Definition eines Moduls

„ Ein Modul ist eine abgeschlossene funktionale Einheit einer Software, bestehend aus einer Folge von Verarbeitungsschritten und Datenstrukturen.

Inhalt eines Moduls ist häufig eine wiederkehrende Berechnung oder Bearbeitung von Daten, die mehrfach durchgeführt werden muss.“

(Wikipedia)

Modulare Programmierung - Allgemeines

Eigenschaften von Modulen

- Module bieten eine Trennung von Schnittstelle und Implementation (Kapselung):
  - Schnittstelle definiert die Datenelemente für die Eingabe und Ausgabe
  - Die Implementierung enthält den eigentlichen Programmcode
- Module können geschachtelt werden (Aufrufhierarchie)
- Vermeidung von redundantem Code
- Strukturierung von großen Softwareprojekten
- Verteilung von Programmierteams auf Module

Modulare Programmierung – Module in C

Abgrenzung zur Objektorientierten Programmierung

- Modularisierung ist keine Objektorientierung
- Module bieten z.B. keine Vererbung / Polymorphie
- Die Glib führt Objektorientierung in C ein
  - Zusätzlich auch andere Funktionen:
    - Komplexe Datenstrukturen
    - Threads
    - Zeitfunktionen
    - Speicherzugriffe u.vm.
  - Details in entsprechendem Vortrag

Modulare Programmierung – Module in C

Vorwort: Deklaration / Definition

Deklaration	Definition

	`char c;`
	`int i,j,k;`
	`double x,y,z;`

`double func(double x);` `extern int a,b,c;`	Definition der Funktionen bzw. Variablen

Modulare Programmierung – Module in C

Vorwort: Geltungsbereich von Variablen und Funktionen

geltungsbereich.c

```
int i,j,k;
double x,y,z;


void func();


extern int a,b,c;

int main()
{
    return 0;
}

void func()
{
    int r = 0;
    Static int i = 0;
}
```

Externe Variablen (in Datei verfügbar)

Funktionsprototyp (immer global verfügbar)

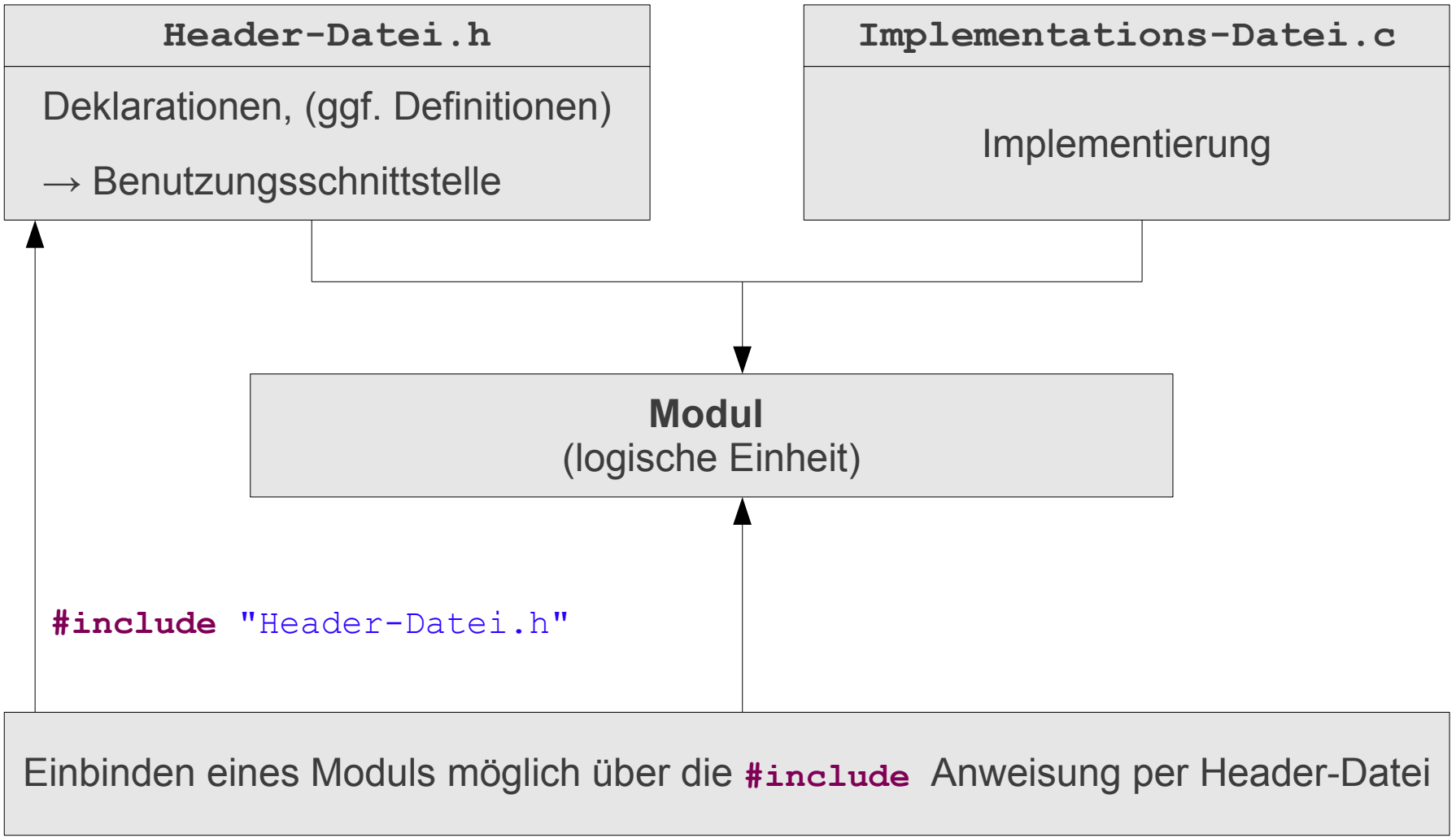
Externe Variablen (über Datei hinaus verfügbar)

a,b,c sind hier nur deklariert, nicht definiert !

Funktionsdefinition

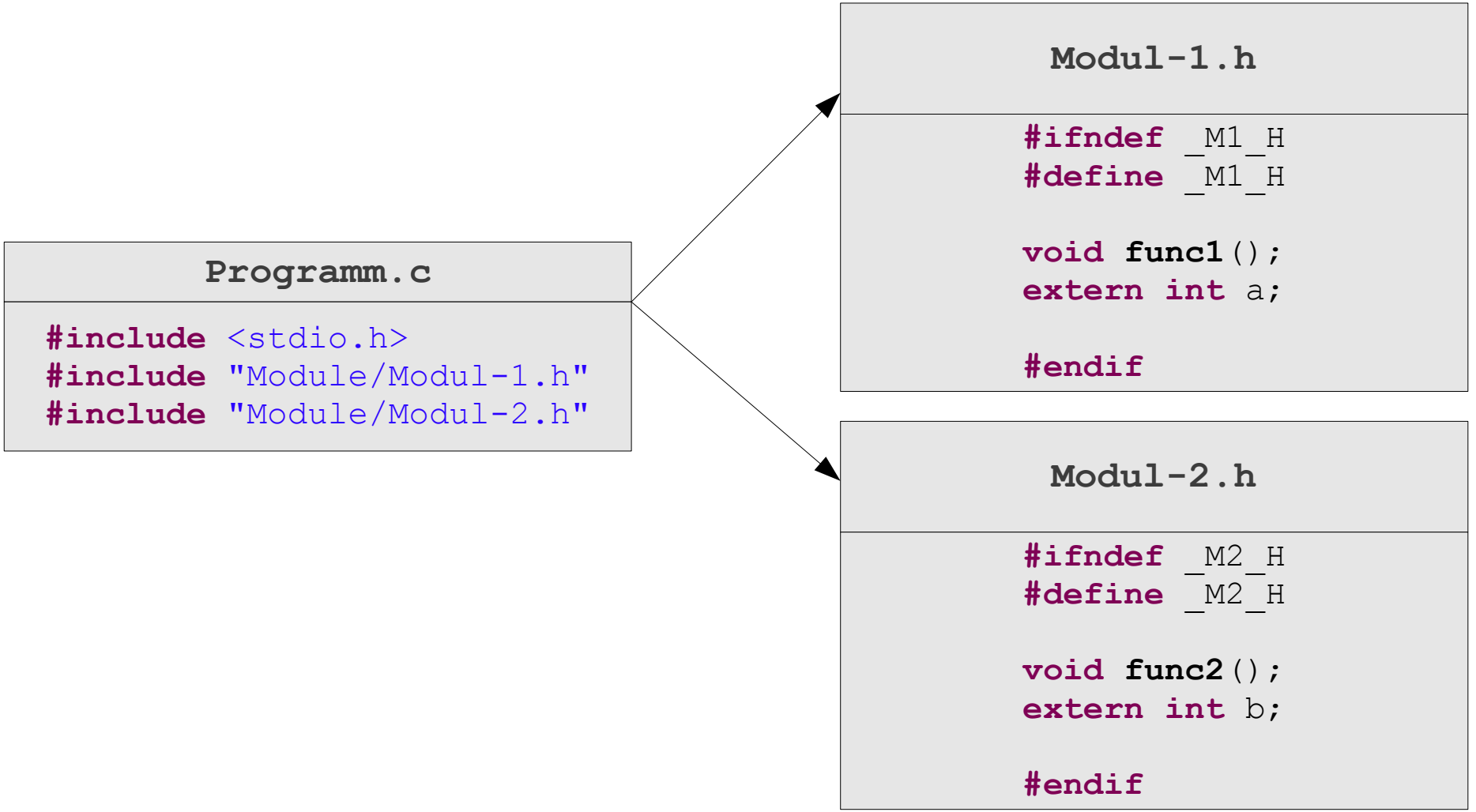
Interne Variable (in Funktion bzw. Block)
Statische Variable (bleibt auch nach Funktions-aufruf bestehen)

Umsetzung von Modulen in C



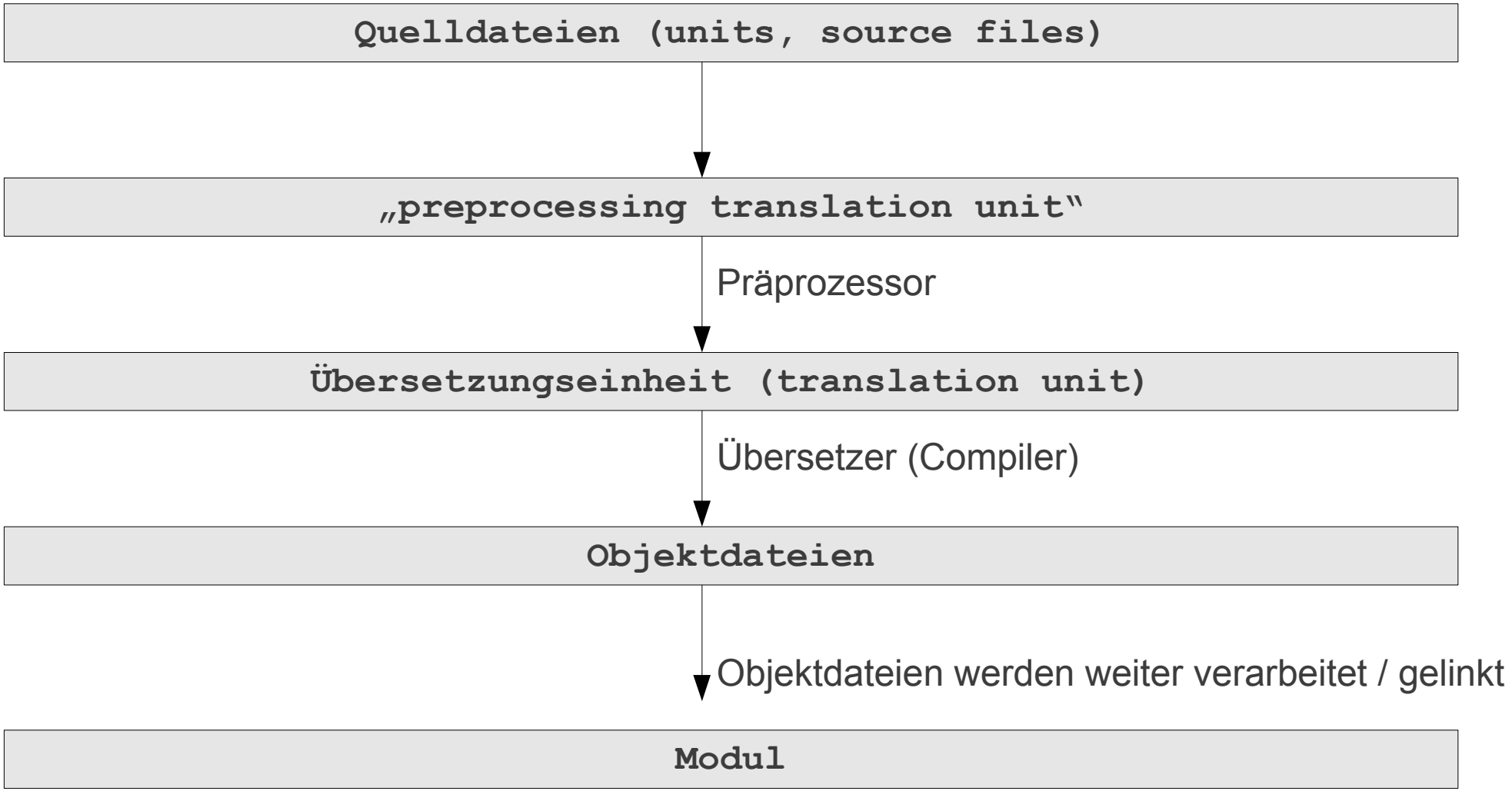
Modulare Programmierung – Module in C

Einbinden von Modulen über bedingtes Include



Modulare Programmierung – Module in C

Exkurs: Übersetzungseinheiten (Compilation units)



Kompilieren von Programmen mit Modulen

- Verschiedene Möglichkeiten:
  - Manuelles Kompilieren & Linken , z.B. mit gcc:
    - gcc -c modul-1.c
    - gcc -c hauptprogramm.c
    - gcc -o Hauptprogramm modul-1.o hauptprogramm.o
  - Verwaltung und Kompilieren durch Makefiles
  - Intelligente Systeme zur Verwaltung
    - z.B. „waf“ (http://code.google.com/p/waf/)
  - Automatisches „Build“ (inkl. Abhängigkeitsprüfung) durch IDE's
    - z.B. Eclipse

Module & Bibliotheken

- Bibliotheken werden wie Module eingebunden
- Für die #include Anweisung gilt:

```
#include <stdio.h>

#include <eigenebibliothek.h>

#include <glib.h>
```

- Der Compiler sucht Bibliotheken im Standard-Pfad
  - Per Parameter können andere Suchpfade angegeben werden (siehe späteres Beispiel)

Modulare Programmierung – Bibliotheken

Motivation

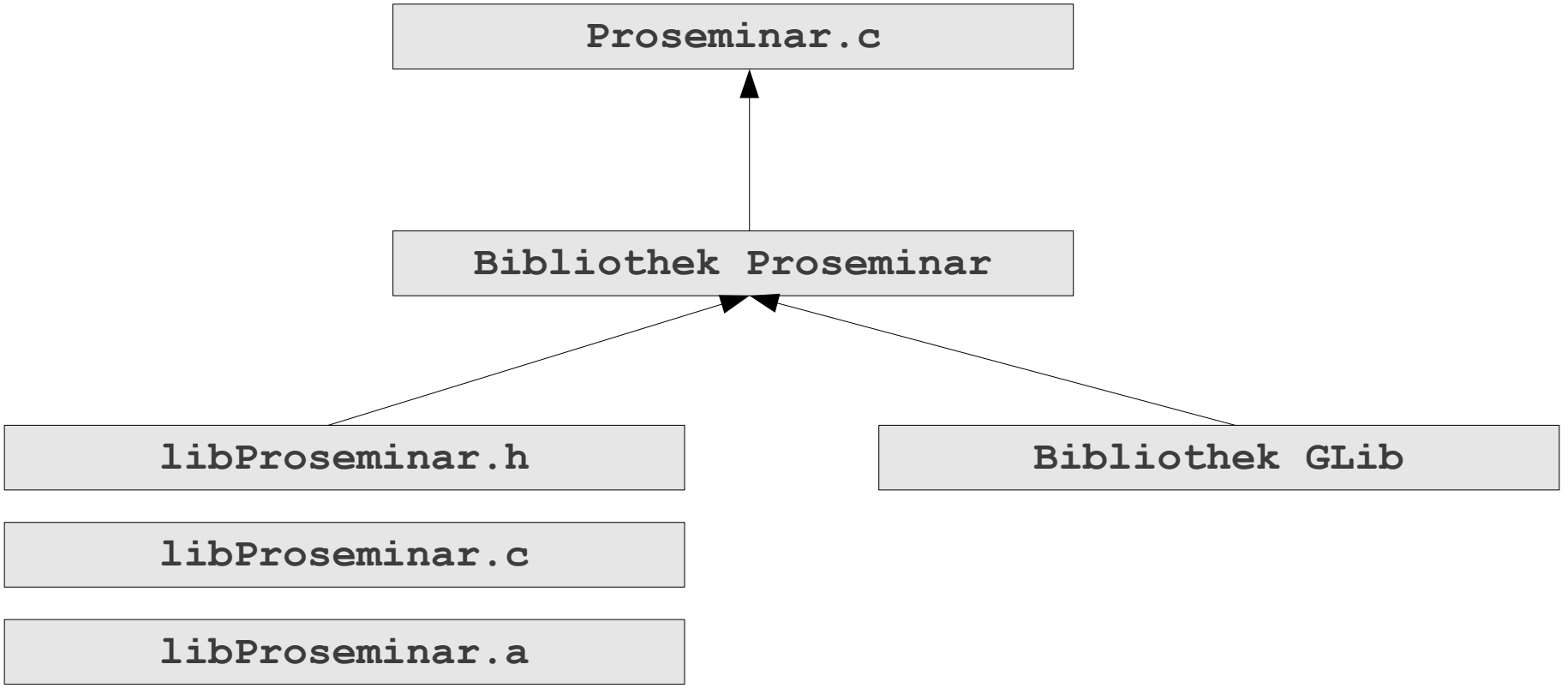
- Geringer Funktionsumfang von C
- Nutzung der Funktionen der C-Standardbibliothek
- Nutzung von eigenen Bibliotheken
- Zentralisierung von Code
  - Dadurch leichtere Wartbarkeit + (Sicherheits-) Updates
- Große Menge an bestehenden Bibliotheken für verschiedenste Anwendungsbereiche

Modulare Programmierung – Bibliotheken

Populäre Bibliotheken

- glibc (GNU C Bibliothek, Implementierung der Std.-Bib.)
- gtk+ (GIMP Toolkit)
- GLib (Gnome Library)
- NAG C Library (Numerische Bibliothek)
- Libwww (Web-API) bzw. libcurl

Anwendungsbeispiel in C - Grundaufbau



Modulare Programmierung – Bibliotheken

Anwendungsbeispiel in C – Code „libProseminar.c“

```
#include <glib.h>
#include <glib/gprintf.h>


void demo()
{
    g_printf(„Benutzen der Glib Funktion g_printf() \n“);
}
```

- Eigene Bibliothek nutzt Funktionen der GLib
- Zum Kompilieren muss die GLib eingebunden werden

Anwendungsbeispiel in C – Übersetzen der Bibliothek

- Einfaches Übersetzen per gcc funktioniert nicht:

```
gcc -o libProseminar.o libProseminar.c
libProseminar.c:1:18: fatal error: glib.h: Datei oder Verzeichnis nicht gefunden
compilation terminated.
```

- Per Parameter muss die Library und der Include Pfad gesetzt werden
- Gängiges Tool: pkg-config

Exkurs: pkg-config

- Tool zum Bereitstellen von Compiler- & Linker-Parametern
- Steht neben Unix z.B. auch für Windows zur Verfügung
- Wichtigste Befehle:
  - pkg-config --list-all
  - pkg-config --libs Bibliothek
  - pkg-config --cflags Bibliothek
  - pkg-config --help

Anwendungsbeispiel in C – Übersetzen der Bibliothek

- Übersetzen mit Hilfe von pkg-config

```
gcc -c -o libProseminar.o libProseminar.c `pkg-config --cflags glib-2.0`
```

- Erzeugt die Objektdatei der Bibliothek

- Die Objektdatei kann dann in ein vorkompiliertes Archiv gewandelt werden (Statische Bibliothek):

```
ar -rcs libProseminar.a libProseminar.o
```

Anwendungsbeispiel in C – Übersetzen des Programms

- Quellcode von „Proseminar.c“

```c
#include <libProseminar.h>

int main(void)
{
    demo();
    return 0;
}
```

- Übersetzen mit Hilfe von pkg-config

```
gcc -o Proseminar Proseminar.c -lProseminar -I. -L. `pkg-config --libs glib-2.0`
```

- Ausgabe nach dem Übersetzen:

```
ludwig@ludwig-pc:~/CProg/Proseminar$ ./Proseminar

Benutzen der GLib Funktion g_printf()
```

Anwendungsbeispiel in C – Übersetzen des Programms

- Übersetzen mit Hilfe von pkg-config

```
gcc -o Proseminar Proseminar.c -lProseminar -I. -L. `pkg-config --libs glib-2.0`
```

- Hinweise zu den Parametern:
  - -lBibliothek : Gibt die einzubindende Bibliothek an
  - -I. : Fügt das aktuelle Verzeichnis als Include-Verz. hinzu
  - -L. : Fügt das aktuelle Verzeichnis als Standard-Such-Pfad hinzu
  - `pkg-config...` : Fügt die Ausgabe von pkg-config als Parameter hinzu

Modulare Programmierung – Fazit

Fazit

- Nutzung von Modulen und Bibliotheken ist gängige Praxis
- Leichtere Wartbarkeit / Wiederverwendung von Code
- Beim Übersetzen von Programmen mit Bibliotheken hilft „pkg-config“
- Probleme beim Übersetzen liegen i.d.R. an falschen Pfadangaben des Compilers
- IDE's oder ähnliche Tools helfen beim Verwalten

Modulare Programmierung – Quellen

- Helmke, Isernhagen - „Softwaretechnik in C und C++“, Hanser-Verlag, 2001
- http://de.wikipedia.org/wiki/GTK%2B
- http://de.wikipedia.org/wiki/Glibc
- http://code.google.com/p/waf/
- http://de.wikipedia.org/wiki/Glib
- http://pronix.linuxdelta.de/C/gtk/gtk_C_Kurs_kapitel2.shtml
- http://www.programmerworld.net/resources/c_library.htm
- http://developer.gnome.org/glib
- http://linux.die.net/man/1/pkg-config