Vorlesung

Systemnahe Programmierung in C

—

Grundlagen der systemnahen Programmierung in C

Sommer 2010

SPiC

A Überblick über die Lehrveranstaltung

A.1 Thema: Systemnahe Programmierung in C

- Grundlagen (GSPiC und SPiC)
  - ➤ Grundzüge von Systemarchitekturen
  - ➤ Einführung in die Programmiersprache C
  - ➤ Mikrocontroller-Überblick, AVR-Architektur
  - ➤ Programmiersprache C: Zeiger, Felder, Strukturen
  - ➤ Interruptverarbeitung und Nebenläufigkeit in Programmen
  - ➤ Betriebssystemunterstützung zur Programmausführung
  - ➤ Speichermodelle: virtuelle Adressräume / physikalischer Speicher
- Vertiefung (nur SPiC)
  - ➤ Programmiersprache C: weitere Vertiefung
  - ➤ Systemschnittstelle UNIX/Linux: Dateisystem und Prozesse

A.2 Aufbau der Lehrveranstaltung

1 Vorlesung

- Überblick über grundlegende Konzepte von systemnaher (= Betriebssystem-naher oder Hardware-naher) Programmierung
- Einführung in die Programmiersprache C
- C-Programmierung "auf der nackten Hardware" (am Beispiel AVR-μC)
  - ➤ Abbildung Speicher ↔ Sprachkonstrukte
  - ➤ Interrupts
  - ➤ Nebenläufigkeit
- C-Programmierung "auf einem Betriebssystem" (am Beispiel Linux)
  - ➤ Gegensatz μC-Umgebung - Betriebssystem
  - ➤ Betriebssystem als Ausführungsumgebung für Programme
  - ➤ Abstraktionen und Dienste eines Betriebssystems

2 Übungen

- Praktische Umsetzung des Vorlesungsstoffs anhand von einigen kleinen Programmieraufgaben
- Tafelübungen:
  - ➤ Hinweise zur Durchführung der Übungsaufgaben
  - ➤ erste Anleitung
  - ➤ Besprechung von Lösungen
  - ➤ Betreuung bei der Bearbeitung der Programmieraufgaben am Rechner
- Rechnerübungen:
  - ➤ selbstständige Programmierung
    - – auf einer kleinen Mikrocontroller-Plattform
    - – unter Linux
  - ➤ Umgang mit Entwicklungswerkzeugen und Betriebssystemen zur Programmentwicklung
    - – Editor, Compiler, AVR-Studio, Windows, Linux
  - ➤ Hilfestellung bei Problemen durch Übungsbetreuer

B Organisatorisches

- Zwei Varianten
- **GSPiC** Grundlagen der systemnahen Programmierung in C
  - Vorlesung + Übung
  - 2 SWS / 2,5 ECTS
  - Studienfächer: EEI
- **SPiC** Systemnahe Programmierung in C
  - Vorlesung + Übung
  - 4 SWS / 5 ECTS
  - Studienfächer: **Mechatronik**, Mathematik, Technomathematik
    - Mathematik, Technomathematik:
      Alternativ kann auch Systemprogrammierung (SysProg) belegt werden!
- Vorlesungen 1 und 2 gemeinsam

B.1 Semesterüberblick GSPiC

KW	Mo	Di	Mi	Do	Fr	Themen / Kapitel
16	19.04.	20.04.	21.04.	22.04.	23.04.	*Organisatorisches, Sprachüberblick, Datentypen, Operatoren* *A, B, C, D.1 – D.5*
		VL 1	VL 2			
17	26.04.	27.04.	28.04.	29.04.	30.04.	*Programmaufbau, Kontrollstrukturen* *D.6 – D.7*
			VL 3	A1 (Blink)		
18	03.05.	04.05.	05.05.	06.05.	07.05.	*Funktionen, Programmstruktur und Module* *D.8 – D.9*
			VL 4	A2 (Snake)		
19	10.05.	11.05.	12.05.	13.05.	14.05.	
20	17.05.	18.05.	19.05.	20.05.	21.05.	*Module, Übersetzen und Binden, Mikrocontroller-Programmierung* *D.9 – D.10, E*
			VL 5			
21	24.05.	25.05.	26.05.	27.05.	28.05.	
	Pfingsten/Berg			A3 (LED)		
22	31.05.	01.06.	02.06.	03.06.	04.06.	*Zeiger und Felder* *F.1 – F.11*
			VL 6			

B.1 Semesterüberblick GSPiC (Fortsetzung)

KW	Mo	Di	Mi	Do	Fr	Themen / Kapitel
23	07.06.	08.06.	09.06.	10.06.	11.06.	*Verbund- und Aufzählungstypen* *F.12 – F.16*
			VL 7	A4 (pLED)		
24	14.06.	15.06.	16.06.	17.06.	18.06.	
25	21.06.	22.06.	23.06.	24.06.	25.06.	*Nebenläufigkeit, Betriebssysteme* *G*
			VL 8	A5 (Ampel)		
26	28.06.	29.06.	30.06.	01.07.	02.07.	*Speicherorganisation, Stapelaufbau* *H*
			VL 9			
27	05.07.	06.07.	07.07.	08.07.	09.07.	*Dateisysteme* *I*
			VL 10	A6 (PrintDir)		
28	12.07.	13.07.	14.07.	15.07.	16.07.	
				Wiederholung		
29	19.07.	20.07.	21.07.	22.07.	23.07.	

B.2 Semesterüberblick SPiC

KW	Mo	Di	Mi	Do	Fr	Themen / Kapitel
16	19.04.	20.04.	21.04.	22.04.	23.04.	*Organisatorisches, Sprachüberblick, Datentypen, Operatoren* *A, B, C, D.1 – D.5*
	VL1	VL2				
17	26.04.	27.04.	28.04.	29.04.	30.04.	*Programmaufbau, Kontrollstrukturen* *D.6 – D.7*
		VL3		A1 (Blink)		
18	03.05.	04.05.	05.05.	06.05.	07.05.	*Funktionen, Programmstruktur und Module* *D.8 – D.9*
		VL4		A2 (Snake)		
19	10.05.	11.05.	12.05.	13.05.	14.05.	*Module, Übersetzen und Binden, Mikrocontroller-Programmierung* *D.9 – D.10, E*
		VL5		A3 (LED)		
20	17.05.	18.05.	19.05.	20.05.	21.05.	*Zeiger und Felder* *F.1 – F.11*
		VL6				
21	24.05.	25.05.	26.05.	27.05.	28.05.	
	Pfingsten			A4 (pLED)		
22	31.05.	01.06.	02.06.	03.06.	04.06.	*Verbund- und Aufzählungstypen* *F.12 – F.16*
		VL7		A5 (Ampel)		

B.2 Semesterüberblick SPiC (Fortsetzung)

KW	Mo	Di	Mi	Do	Fr	Themen / Kapitel
23	07.06.	08.06.	09.06.	10.06.	11.06.	*Nebenläufigkeit, Betriebssysteme* *G*
		VL8		A6(PrintDir)		
24	14.06.	15.06.	16.06.	17.06.	18.06.	*Speicherorganisation, Stapelaufbau* *H*
		VL9		A7 (fish)		
25	21.06.	22.06.	23.06.	24.06.	25.06.	*Dateisysteme* *I*
		VL10		A8 (tqsh)		
26	28.06.	29.06.	30.06.	01.07.	02.07.	*Prozesse, Prozesszustände, Prozesswechsel, Prozesserzeugung, Ausführen von Programmen* *J.1 – J.6*
		VL11		A9 (find_max)		
27	05.07.	06.07.	07.07.	08.07.	09.07.	*Signale* *J.7*
28	12.07.	13.07.	14.07.	15.07.	16.07.	*Threads, Koordinierung* *J.8 – J.9*
		VL13		Wdh		
29	19.07.	20.07.	21.07.	22.07.	23.07.	*pthreads, Threads in Java* *J.10 – J.11*

B.3 Vorlesungsbetrieb

- Dozenten
  - Jürgen Kleinöder (SPiC, GSPiC)
  - Daniel Lohmann (GSPiC, SPiC)
- SPiC-Webseite: `www4.informatik.uni-erlangen.de/Lehre/SS10/V_SPIC/`
- GSPiC-Webseite: `www4.informatik.uni-erlangen.de/Lehre/SS10/V_GSPIC/`

B.4 Vorlesungsskript

- Vorlesungsfolien
  - im pdf-Format auf der Webseite
  - Gutscheinverkauf zum Bezug von Folienkopien, Schutzgebühr 1 EUR
    - ➤ Kopien werden jeweils vor der Vorlesung ausgegeben

B.5 Literatur

- Literatur
  - zu der Programmiersprache C
    - ➤ Peter A. Darnell, Philip E. Margolis:
      C: A Software Engineering Approach, 3. Edition, Springer, 1996.
    - ➤ Karlheinz Zeiner:
      Programmieren lernen mit C, 2. Auflage, Carl Hanser, 1996.
    - ➤ B. W. Kernighan, D. M. Ritchie:
      Programmieren in C, 2. Auflage, Carl Hanser, 1990.

B.6 Übungen

- Beginn: Do. 29.04.2009 (Übungswoche jeweils Do. - Mi.)
- Tafelübungen (teilweise auch "am Rechner")
  - Erläuterung zur Benutzung der Rechnerumgebung
  - Anleitung zu den Aufgabenstellungen
  - Besprechung der Lösungen
- Rechnerübungen Raum 01.155 Informatik-Hochhaus
  - selbstständige Bearbeitung von Aufgaben
  - Übungsleiter stehen bei Fragen und Problemen zur Verfügung
- Verantwortlich
  - Moritz Strübe (SPiC, GSPiC), Wanja Hofer (GSPiC, SPiC)
  - Übungsbetreuer: Rainer Müller, Tobias Scharpff, Martin Klüpfel, Fabian Festerra, Franziska Bertelshofer, Christian Schlumberger, Sebastian Schinabeck

B.6 Übungen (2)

- Übungsbetrieb für **SPiC**
  (Mechatronik-Bachelor, 2. Semester, Mathematik, Technomathematik)
  - 9 Gruppen zur Auswahl
    - ➤ Dauer einer Übung 90 Minuten
    - ➤ maximale Teilnehmerzahl 12 Personen
  - Termine **Tafelübung** (Änderungen vorbehalten):
    Mo. 14-16, 16-18
    Di. 12-14, 16-18
    Mi. 10-12, 14-16
    Do. 14-16, 16-18
    Fr. 10-12
  - Termine **Rechnerübung**:
    Mo. 12 - 14
    Di. 14 - 16
    Mi. 08 - 10, 14 - 16
    Do. 10 - 12, 16 - 18

B.6 Übungen (3)

- Übungsbetrieb für **GSPiC**
  (EEI-Bachelor, 2. Semester)
  - 8 Gruppen zur Auswahl
    - ➤ Dauer einer Übung 90 Minuten
    - ➤ maximale Teilnehmerzahl 12 Personen
  - Termine (Änderungen vorbehalten):
    Mo. 10-12, 12-14
    Di. 14-16, 16-18
    Mi. 14-16
    Do. 8-10, 12-14
    Fr. 14-16
  - **Tafel-** und **Rechnerübung** im Wechsel

B.6 Übungen (4)

- Anmeldung zu den Tafelübungen
  - ➤ heute (Di, 20.4.) ab 16:00 (nach der Vorlesung)
  - ➤ über Web-Anmeldesystem "waffel"
  - ➤ Link auf der Übungs-Webseite
  - ➤ Bei der Anmeldung Auswahl des Tafelübungstermins

B.7 Programmieraufgaben

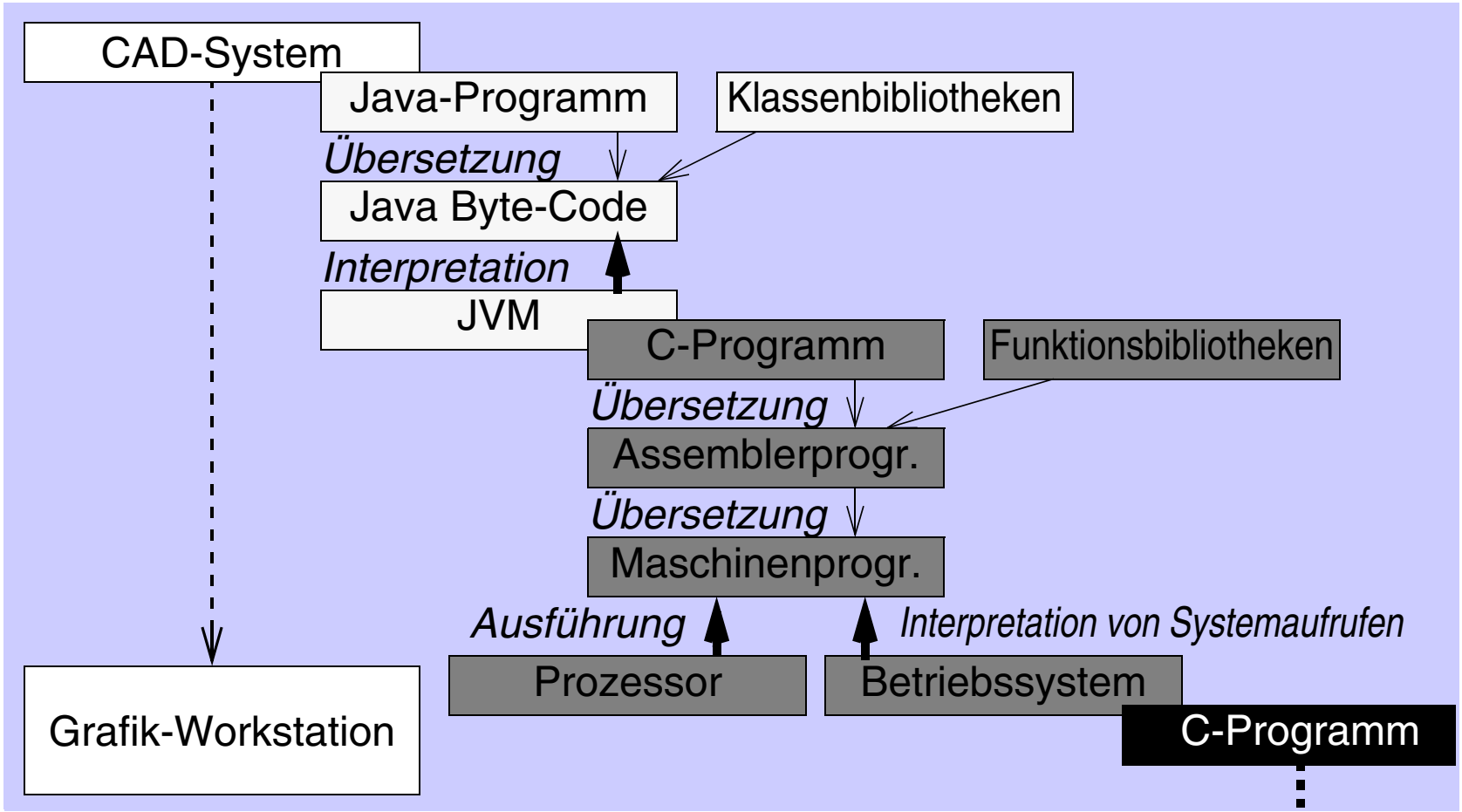
- Programmieraufgaben teilweise alleine, teilweise in 2er-Gruppen zu bearbeiten
- Lösungsaufgaben mit Abgabeskript am Rechner abgeben
- Lösung wird durch Skripte überprüft;
  zusätzlich korrigieren wir die Abgaben und geben sie zurück;
  außerdem geben wir Hinweise auf typische Fehler
  in der Vorlesung und den Tafelübungen.
- ★ abgegebene Aufgaben werden bepunktet
  durch die Punkte auf Übungsaufgaben können bis zu
  10 % Bonuspunkte
  bei der Prüfungsklausur erarbeitet werden
  - ➤ Voraussetzung: abgegebene Aufgaben müssen jederzeit in den Tafelübungen vorgestellt werden können
    (impliziert Anwesenheit!)

B.8 Prüfung

- Prüfung (Klausur)
  - ➤ Termin voraussichtlich Ende Juli/Anfang August
    Dauer GSPiC: 60 min, SPiC: 90 min
  - ➤ Inhalt: Fragen zum Vorlesungsstoff + Programmieraufgabe

C Systemarchitekturen

- Große Diskrepanz zwischen Anwendungsproblem und dem Ablauf der Lösung auf einer Hardware



C.1 Softwareschichten

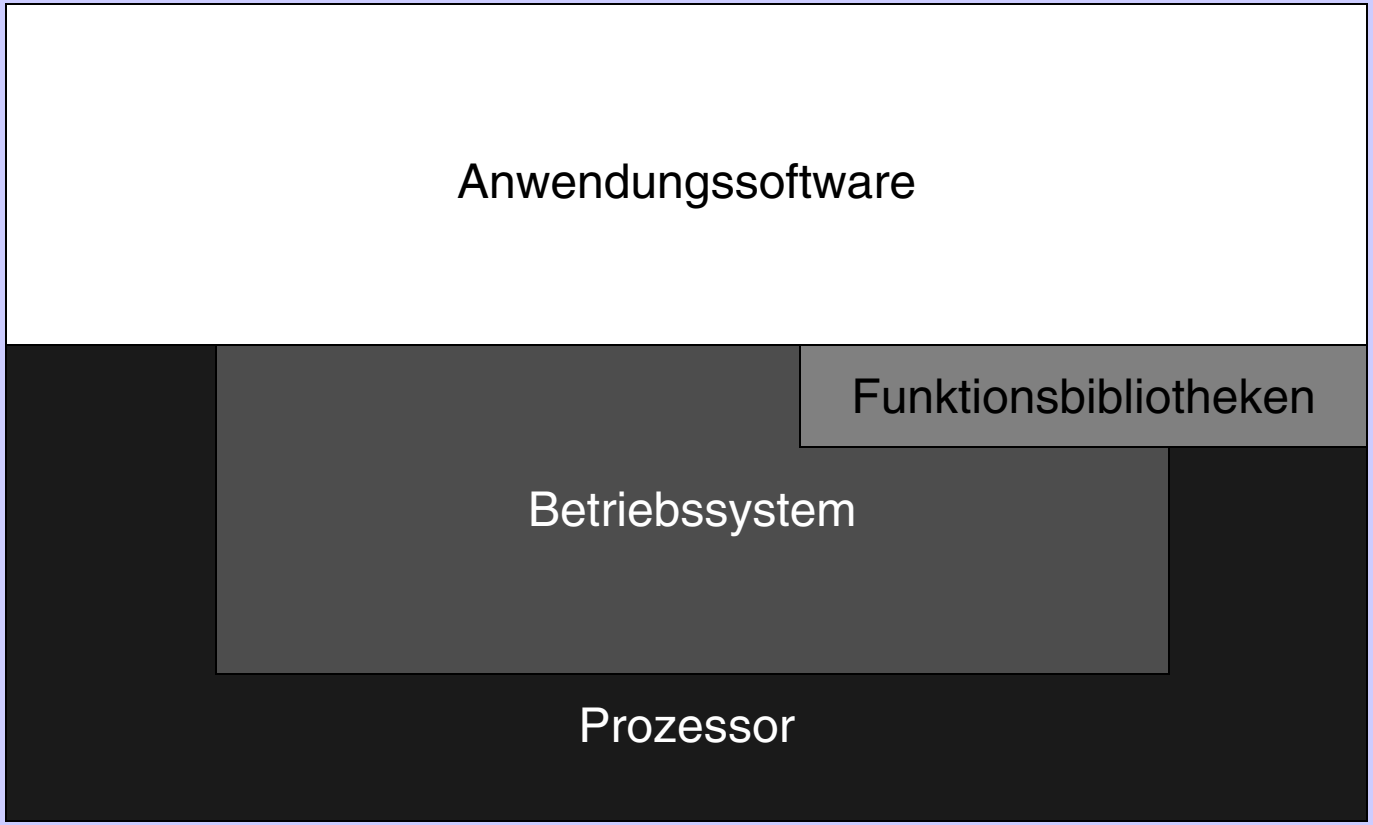
- Anwendungs-/Problemorientierte Darstellungen
  - ◆ Modelle
    - ➤ Matlab/Simulink
    - ➤ UML
  - ◆ Programmiersprachen / höhere Abstraktionsebenen
    - ➤ Software-Komponenten
    - ➤ Java, C#, C++, Tcl/TK
- Softwarewerkzeuge konvertieren / generieren
  - ➤ Matlab/Simulink ➔ C
- Ausführungsumgebungen unterstützen / konvertieren / interpretieren
  - ➤ Enterprise Java Beans
  - ➤ JVM oder .NET

➥ Ziel: durch Prozessor ausführbarer Maschinencode

C.1 Softwareschichten (2)

- verschiedene Ausführungsmodelle für Maschinencode
  - vollständig durch den Prozessor ausführbar
    - ➤ alle Funktionen müssen vollständig durch die Werkzeuge in direkt ausführbaren Maschinencode umgewandelt worden sein
    - ➤ keinerlei weitere Unterstützung zur Laufzeit erforderlich
    - ➤ kann so in ROM oder EPROM gespeichert werden
    - ➤ z. B. Steuerung einer Waschmaschine
  - zusätzliche Unterstützung zur Ausführungszeit erforderlich
    - ➤ "darunter liegende" Softwareschicht realisiert Dienste: Betriebssystem
    - ➤ z. B. Daten in Datei speichern, Daten über Internet übertragen
    - ➤ Realisierung: partielle Interpretation
      bestimmte Maschinencodes werden nicht direkt vom Prozessor ausgeführt sondern stoßen die Abarbeitung von Betriebssystemfunktionen an

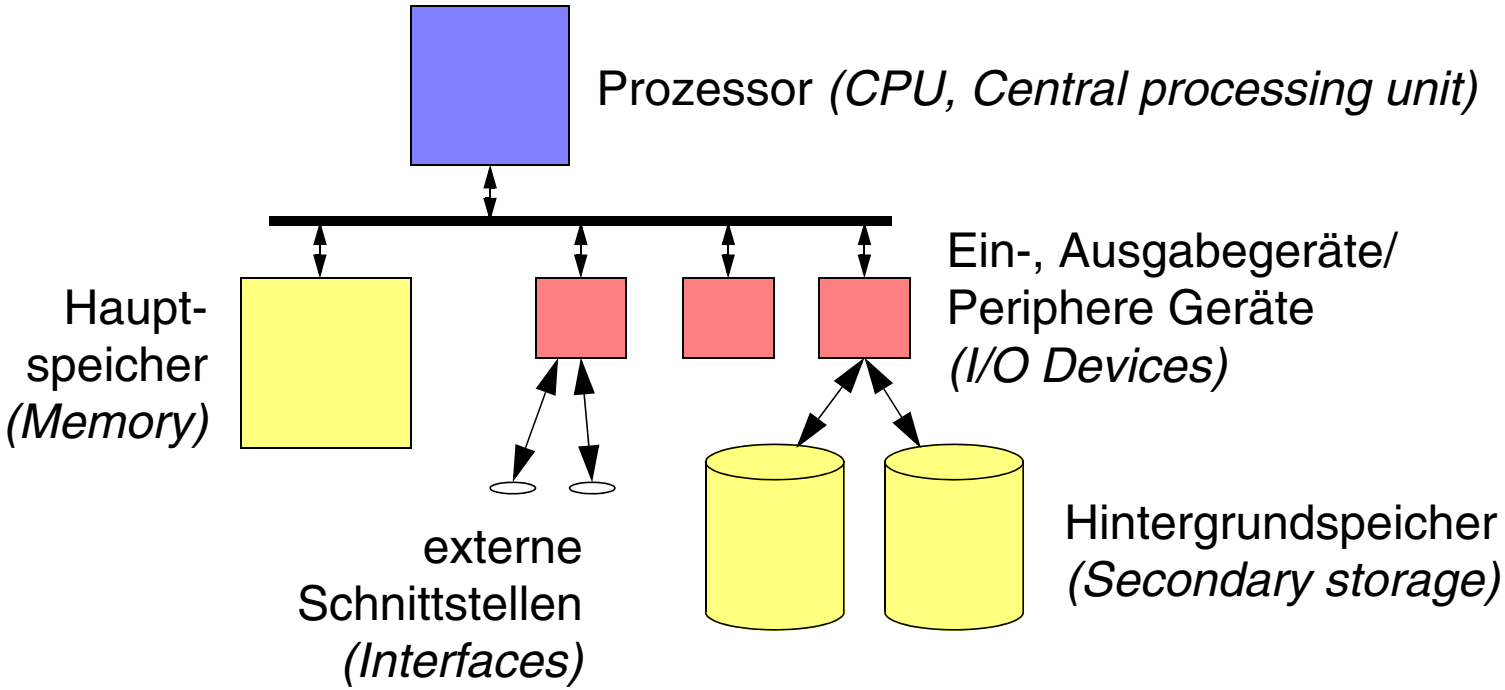
C.1 Softwareschichten (3)



C.2 Was sind Betriebssysteme?

- DIN 44300
  - „...die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die **Basis der möglichen Betriebsarten** des digitalen Rechensystems bilden und die insbesondere die **Abwicklung von Programmen steuern und überwachen.**“
- Andy Tanenbaum
  - „...eine Software-Schicht ..., die alle Teile des Systems verwaltet und dem Benutzer eine Schnittstelle oder eine *virtuelle Maschine* anbietet, die einfacher zu verstehen und zu programmieren ist [als die nackte Hardware].“
- ★ Zusammenfassung:
  - Software zur Verwaltung und Virtualisierung der Hardwarekomponenten (Betriebsmittel)
  - Programm zur Steuerung und Überwachung anderer Programme

1 Verwaltung von Betriebsmitteln



1 Verwaltung von Betriebsmittel (2)

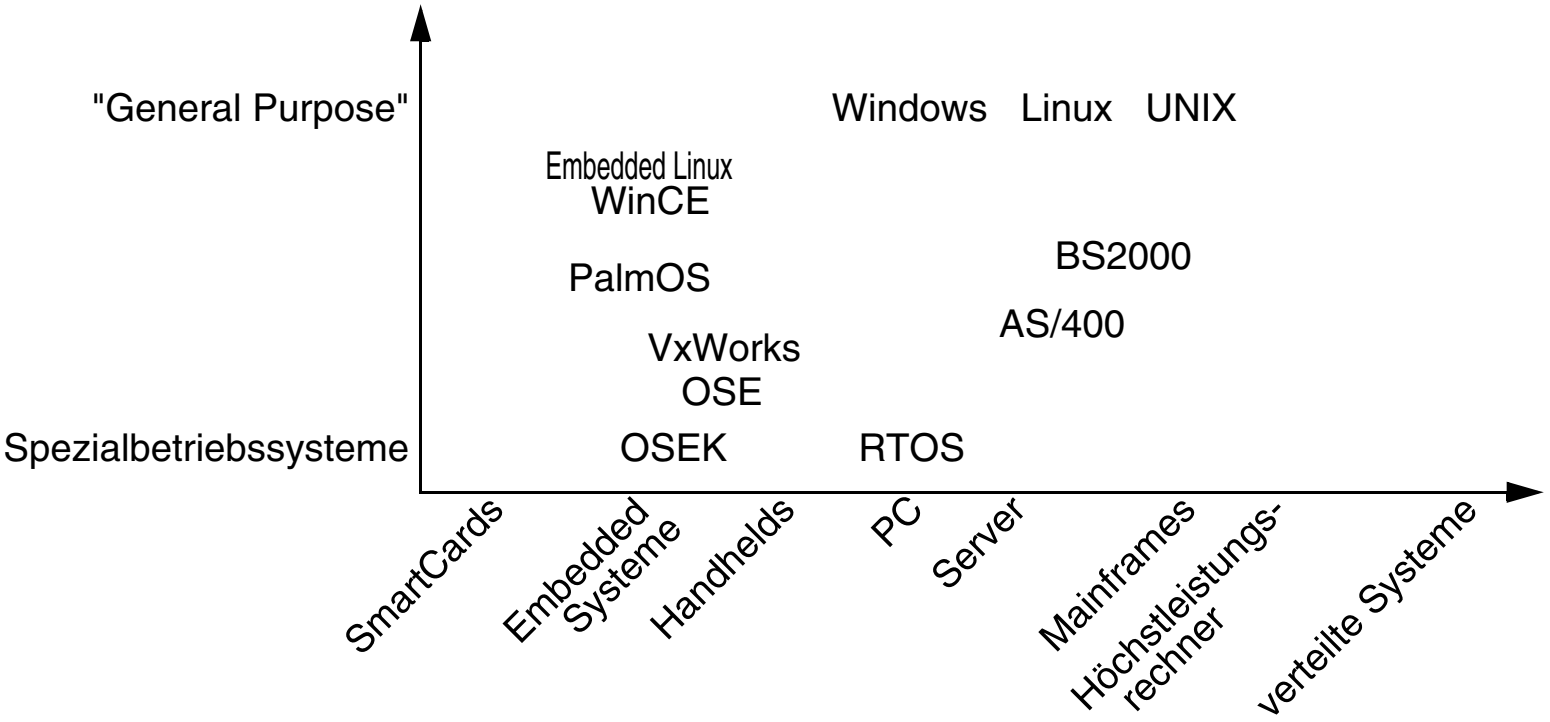
- Resultierende Aufgaben
  - Multiplexen von Betriebsmitteln für mehrere Benutzer bzw. Anwendungen
  - Schaffung von Schutzumgebungen
  - Bereitstellen von Abstraktionen zur besseren Handhabbarkeit der Betriebsmittel

- Ermöglichen einer koordinierten gemeinsamen Nutzung von Betriebsmitteln, klassifizierbar in
  - aktive, zeitlich aufteilbare (Prozessor)
  - passive, nur exklusiv nutzbare (periphere Geräte, z.B. Drucker u.Ä.)
  - passive, räumlich aufteilbare (Speicher, Plattenspeicher u.Ä.)

- Unterstützung bei der Fehlererholung

2 Klassifikation von Betriebssystemen

- Unterschiedliche Klassfikationskriterien
  - ➤ Zielplattform
  - ➤ Einsatzzweck, Funktionalität



2 Klassifikation von Betriebssystemen (2)

- Wenigen "General Purpose"- und Mainframe/Höchstleistungsrechner-Betriebssystemen steht eine Vielzahl kleiner und kleinster Spezialbetriebssysteme gegenüber:

  C51, C166, C251, CMX RTOS, C-Smart/Raven, eCos, eRTOS, Embos, Ercos, Euros Plus, Hi Ross, Hynet-OS, LynxOS, MicroX/OS-II, Nucleus, OS-9, OSE, OSEK Flex, OSEK Turbo, OSEK Plus, OSEKtime, Precise/MQX, Precise/RTCS, proOSEK, pSOS, PXROS, QNX, Realos, RTMOSxx, Real Time Architect, ThreadX, RTA, RTX51, RTX251, RTX166, RTXC, Softune, SSXS RTOS, VRTX, VxWorks, …

➥ Einsatzbereich: Eingebettete Systeme, häufig Echtzeit-Betriebssysteme, über 50% proprietäre (in-house) Lösungen

- Alternative Klassifikation: nach Architektur

3 Betriebssystemarchitekturen

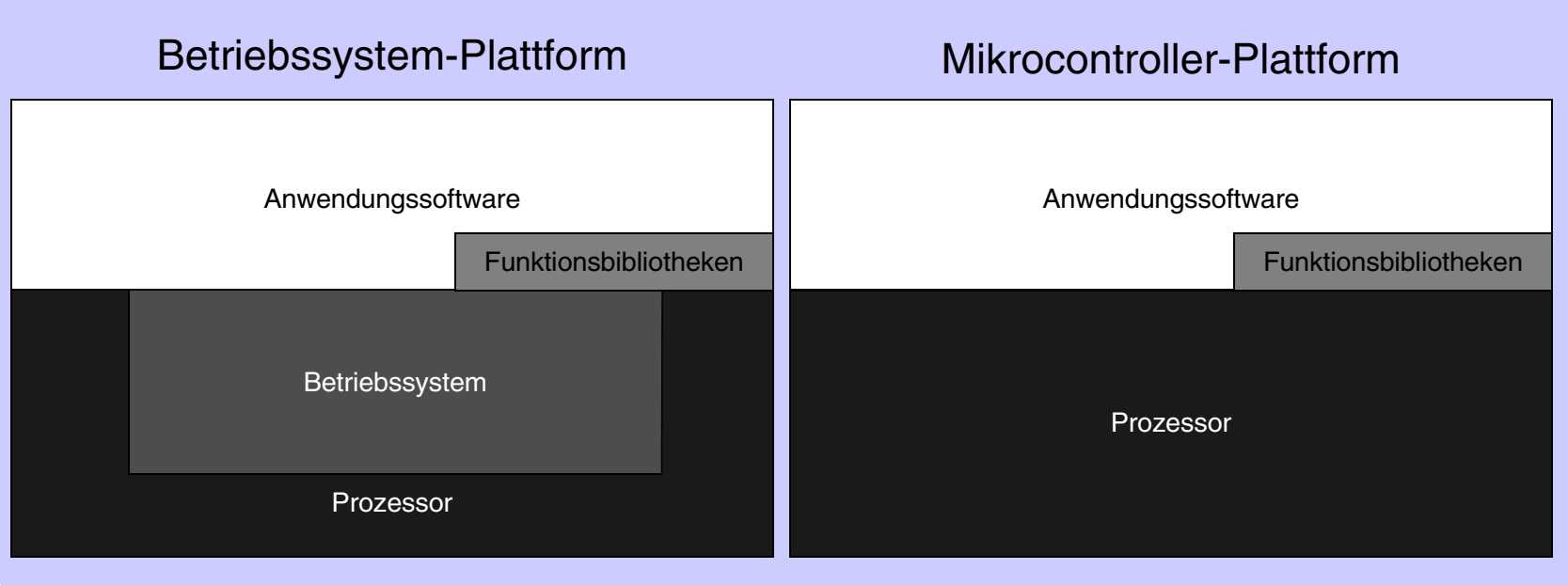
- Umfang zehntausende bis mehrere Millionen Befehlszeilen
  - Strukturierung hilfreich
- Verschiedene Strukturkonzepte
  - monolithische Systeme
  - geschichtete Systeme
  - Minimalkerne
  - Laufzeitbibliotheken (minimal, vor allem im Embedded-Bereich)
- Unterschiedliche Schutzkonzepte
  - ➤ kein Schutz
  - ➤ Schutz des Betriebssystems
  - ➤ Schutz von Betriebssystem und Anwendungen untereinander
  - ➤ feingranularer Schutz auch innerhalb von Anwendungen

4 Betriebssystemkomponenten

- Speicherverwaltung
  - Wer darf wann welche Information wohin im Speicher ablegen?
- Prozessverwaltung
  - Wann darf welche Aufgabe bearbeitet werden?
- Dateisystem
  - Speicherung und Schutz von Langzeitdaten
- Ein/Ausgabe
  - Kommunikation mit der "Außenwelt" (Benutzer/Rechner)

C.3 Mikrocontroller vs. Betriebssystem-Plattform

- Entscheidende Unterschiede:
  - Betriebssystem-Unterstützung entfällt



  - Prozessor bietet in der Regel weniger / andere Funktionalität
    - ➤ kein virtueller Speicher
    - ➤ kein Speicherschutz
    - ➤ einfachere Peripherie-Ansteuerung

D Einführung in die Programmiersprache C

D.1 C vs. Java

- Java: objektorientierte Sprache
  - ➤ zentrale Frage: aus welchen Dingen besteht das Problem
  - ➤ Gliederung der Problemlösung in Klassen und Objekte
  - ➤ Hierarchiebildung: Vererbung auf Klassen, Teil-Ganze-Beziehungen
  - ➤ Ablauf: Interaktion zwischen Objekten
- C: imperative / prozedurale Sprache
  - ➤ zentrale Frage: welche Aktivitäten sind zur Lösung des Problems auszuführen
  - ➤ Gliederung der Problemlösung in Funktionen
  - ➤ Hierarchiebildung: Untergliederung einer Funktion in Teilfunktionen
  - ➤ Ablauf: Ausführung von Funktionen

D.1 C vs. Java

1 C hat nicht

- Klassen und Vererbung
- Objekte
- umfangreiche Klassenbibliotheken

2 C hat

- Zeiger und Zeigerarithmetik
- Präprozessor
- Funktionsbibliotheken

D.2 Sprachüberblick

1 Erstes Beispiel (C-Programm unter Linux)

- Die Datei `hello.c` enthält die folgenden Zeilen:

```
#include <stdio.h>
/* say "hello, world" */
int main()
{
    printf("hello, world\n"); return 0;
}
```

- Die Datei wird mit dem Kommando `cc` übersetzt:

```
% cc hello.c
oder
% gcc hello.c
```

(C-Compiler)

(GNU-C-Compiler)

es entsteht eine Datei `a.out`, die das ausführbare Programm enthält.

➤ ausführbares Programm liegt in Form von Maschinencode des Zielprozessors vor (kein Byte- oder Zwischencode)!

1 Erstes Beispiel (2)

- Mit der Option `-o` kann der Name der Ausgabedatei auch geändert werden – z. B.

```
% cc -o hello hello.c
```

- Das Programm wird durch Aufruf der Ausgabedatei ausgeführt:

```
% ./hello
hello, world
%
```

- Kommandos werden so in einem Fenster mit UNIX/Linux-Kommandointerpreter (Shell) eingegeben
  - ➤ es gibt auch integrierte Entwicklungsumgebungen (z. B. Eclipse)

2 Erstes Beispiel (C-Programm für AVR-Mikrocontroller)

- Die Datei `red.c` enthält die folgenden Zeilen:

```
/* switch red led on */
#include <led.h>
void main()
{
    sb_led_on(RED0); while(1);
}
```

- Die Datei wird mit dem Kommando `avr-gcc` übersetzt:

```
% avr-gcc -o red.elf -ffreestanding -mmcu=atmega32 ... red.c
```

  - ➤ im Gegensatz zur Übersetzung eines Programms für Linux muss hier
    - – die Zielplattform angegeben werden *(Cross-Compilation)*
    - – Angaben über Bibliotheken und include-Dateien angegeben werden (...)

- Vereinfachung über "Makefile"
  - ➤ Details in der Übung

2 Erstes Beispiel (C-Programm für AVR-Mikrocontroller) (2)

- In der Datei `red.elf` liegt der ausführbare Programmcode für den Mikrocontroller vor
- dieser muss anschliessend auf den Mikrocontroller geladen werden
  - ➤ Mikrocontroller über USB-Schnittstelle an Entwicklungs-PC anschließen
  - ➤ Programm zum Übertragen des Codes (*Flashen*) starten

```
% make -f /proj/i4spic/pub/debug.mk red.elf.flash
```

- Weitere Details in den Übungen!

3 Aufbau eines C-Programms

- frei formulierbar **- *Zwischenräume*** *(Leerstellen, Tabulatoren, Newline und Kommentare*) werden i. a. ignoriert - sind aber zur eindeutigen Trennung direkt benachbarter Worte erforderlich

- ***Kommentar*** wird durch `/*` und `*/` geklammert
  keine Schachtelung möglich

- ***Identifier*** (Variablennamen, Marken, Funktionsnamen, ...) sind aus Buchstaben, gefolgt von Ziffern oder Buchstaben aufgebaut
  - "_" gilt hierbei auch als Buchstabe
  - Schlüsselwörter wie `if`, `else`, `while`, usw. können nicht als *Identifier* verwendet werden
  - ***Identifier*** müssen vor ihrer ersten Verwendung ***deklariert*** werden

- Anweisungen werden generell durch `;` abgeschlossen

4 Allgemeine Form eines C-Programms:

```
/* globale Variablen */
    ...

/* Hauptprogramm */
main(...)
{
    /* lokale Variablen */
        ...
    /* Anweisungen */
        ...
}

/* Unterprogramm 1 */
function1(...)
{
    /* lokale Variablen */
        ...
    /* Anweisungen */
        ...
}

/* Unterprogramm n */
functionN(...)
{
    /* lokale Variablen */
        ...
    /* Anweisungen */
        ...
}
```

5 Wie ein C-Programm nicht aussehen sollte:

```
#define o define
#o ___o write
#o ooo (unsigned)
#o o_o_ 1
#o _o_ char
#o _oo goto
#o _oo_ read
#o o_o for
#o o_ main
#o o__ if
#o oo_ 0
#o _o(_,__,___)(void)___o(_,__,ooo(___))
#o __o(o_o_<<((o_o_<<(o_o_<<o_o_))+(o_o_<<o_o_)))
+(o_o_<<(o_o_<<(o_o_<<o_o_)))
o_(){_o_ _=oo_,__,___,____[__o];_oo _______;_____:___=__o-o_
_______:
_o(o_o_,____,__=(_-o_o_<___?_-
o_o_:___));o_o(;__;_o(o_o_,"\b",o_o_),__--);
_o(o_o_," ",o_o_);o__(--___)_oo
_______;_o(o_o_,"\n",o_o_);______:o__(_=_oo_(
oo_,____,__o))_oo _____;}
```

sieht eher wie Morse-Code aus, ist aber ein **gültiges** C-Programm.

D.3 Datentypen

- Datentypen
  - ➤ Konstanten
  - ➤ Variablen

- ◆ Ganze Zahlen
- ◆ Fließkommazahlen
- ◆ Zeichen
- ◆ Zeichenketten

1 Was ist ein Datentyp?

- Menge von Werten
  +
  Menge von Operationen auf den Werten

  - **Literale** — Darstellung für einen konkreten Wert (`2`, `3.14`, `'a'`)
  - **Variablen** — Namen für Speicherplätze, die einen Wert aufnehmen können

  ➥ Literale und Variablen besitzen einen Typ

- Datentypen legen fest:
  - Repräsentation der Werte im Rechner
  - Größe des Speicherplatzes für Variablen
  - erlaubte Operationen

- Festlegung des Datentyps
  - implizit durch Verwendung und Schreibweise (Zahlen, Zeichen)
  - explizit durch **Deklaration** (Variablen)