

Systemprogrammierung

Prof. Dr. Aris Christidis

WS 2014 / 15

- SP-Übungen: 2. + 3. Übung obligatorisch
- Programmieren in C mit MS Visual Studio 2010
- Eigener Name als erste Zeile der Programmausgabe (`printf`)
- In allen Quellen: Name & Matr.Nr. im Kopf als Kommentar
- Vorführung, dann Abgabe: Dateien *.h, *.c, *.exe, *.vcxproj – vorzugsw. in Verzeichnis-Struktur. Bitte nicht zumüllen!
- Per Email gezippt (z.B. Uebg3.zip) bis zum Abgabetermin einsenden (auch an Tutor! 5 Wochen – s.a. www)
- Keine (positiven) Benachrichtigungen vorgesehen!

- Übungsblätter, frühere Klausuren (inkl. Lösungshilfe) Materialien, Programme, Vorlesungsfolien (KEIN Skript):
<http://homepages.thm.de/christ/>
- MSDN Library for Visual Studio (Microsoft Developer Network)
- Regionales Rechenzentrum für Niedersachsen (RRZN) / Softwareberatung Herdt:
„Die Programmiersprache C. Ein Nachschlagewerk“,
RRZN, 18. Auflage 2010, € 3,50 1995
(http://www.rrzn.uni-hannover.de/buecher.html?&no_cache=1)
Bezug: Hochschulrechenzentrum der Justus-Liebig-Universität Gießen
Heinrich-Buff-Ring 44, 35392 Gießen, Tel. 0641/99-13600
Wegbeschreibung: <http://www.uni-giessen.de/hrz/organisation/weg.html>



Literatur

- B.W. Kernighan, D.M. Ritchie:
„Programmieren in C“
(2.Ausgabe, ANSI-C), Carl Hanser 1990
€ 32,90

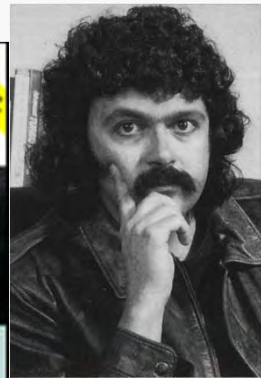
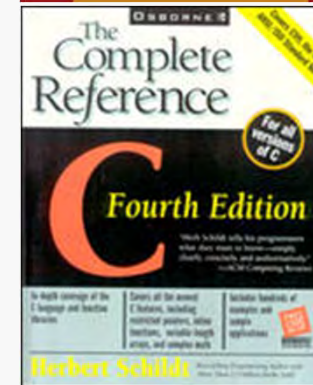
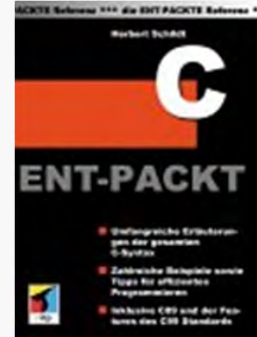
<http://bookzz.org>

- H.Schildt:
„C-The Complete Reference“, 4th Ed.
McGraw-Hill 2000, \$39,99

„C Ent-Packt“, mitp, 2001, €39,95

www.mitp.de/imperia/md/content/vmi/0732/0732_listings.zip

- Ph. A. Laplante:
"Real-Time Systems – Design and Analysis" 1st Ed.
IEEE Press 2004, \$105,00



- J.Nehmer, P.Sturm:
„Systemsoftware – Grundlagen moderner Betriebssysteme“
dpunkt-Verlag 2001
- P. Pepper:
„Grundlagen der Informatik“
(2. Auflage), R.Oldenbourg 1995
(nur noch in der Bibliothek: nicht mehr aufgelegt)
- Andrew S. Tanenbaum:
„Modern Operating Systems“
(2nd Edition) Prentice Hall 2001 – bzw.:
„Moderne Betriebssysteme“
(2. Auflage) Prentice Hall 2002, €49,95



- Ch. Petzold:
„Windows Programmierung“
(5.Auflage) Microsoft Press 2000, €59,90
- D. A. Solomon, M. Russinovich:
„Inside Microsoft Windows 2000“
(3. Auflage) Microsoft Press 2000, €66,00
- J. M. Richter:
„Microsoft Windows Programmierung für Experten“
Microsoft Press Deutschland 2000, €119,90



- div. Internet-Sites (t.b.c.)

Inhalt dieser Vorlesung:

Aspekte* der Struktur u. Funktionsweise
von Sw-Plattformen und -Umgebungen

(*) Beispielhafte
Sw-Techniken

- Systeme, Programmierung, Systemprogramme, SysProg
Betriebssysteme, GUIs, Timer, Callbacks, Globale Variablen
- Prozeß-Interaktion u. -Kommunikation
Blockierungsarten, Vermeidungsstrategien, Semaphore (dinier. Phil.)
- Ereignisse und Fenstersysteme
GUI-Design und -Philosophie, Widgets, openSource-Beispiele
- Exemplarische Betrachtung von Windows
Thread-Programmierung ⇔ Windows-Programmierung

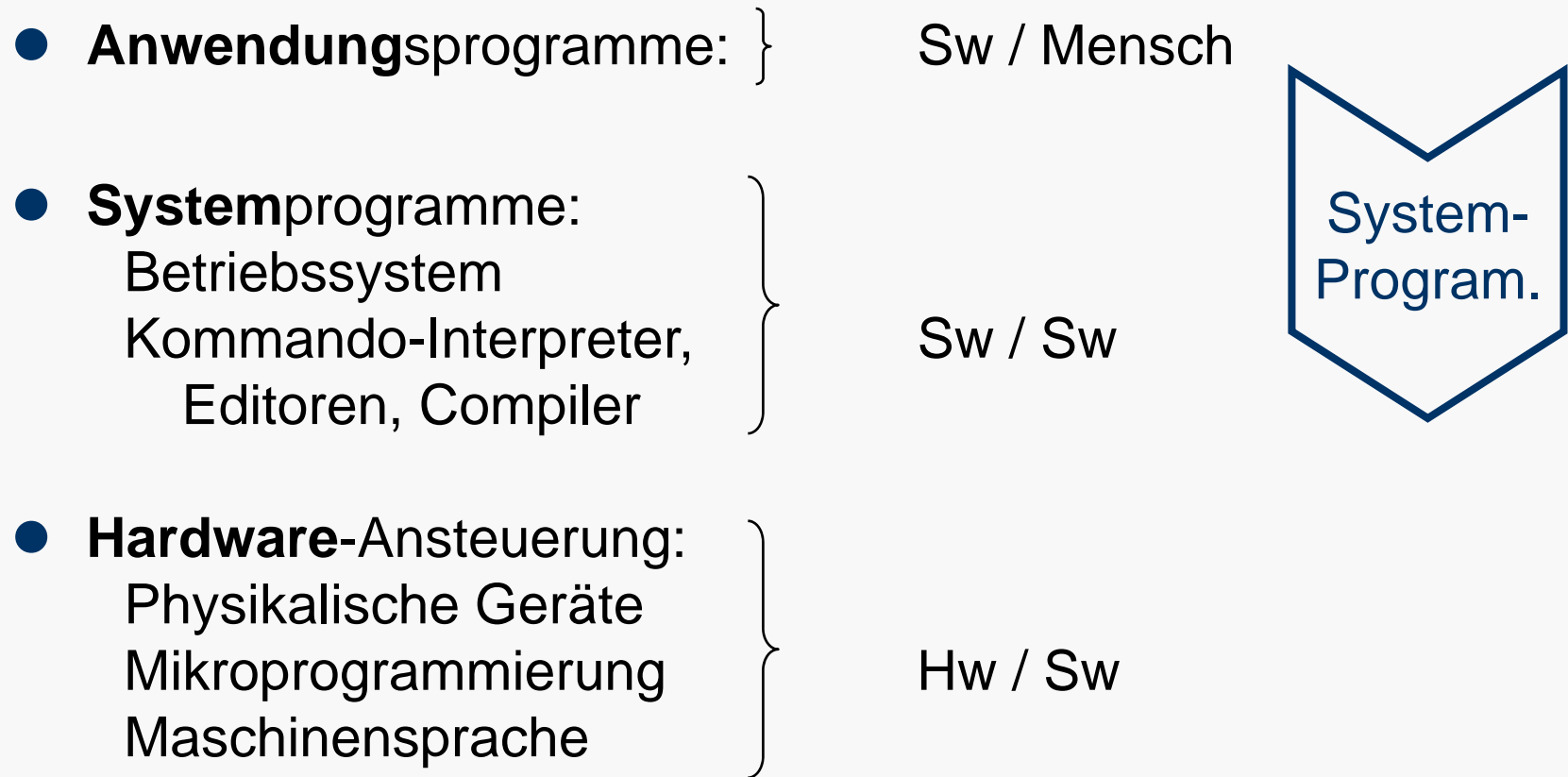
Hier verwendete Begriffe:

- **Sw-Plattform:**
Die ‚tiefste‘ (Hw-nächste) genutzte Sw-Schicht
- i.d.R. das Betriebssystem
- **Sw-Umgebung:**
Satz von Programmen, die für eine bestimmte Nutzung die Kommunikation mit der Plattform übernehmen
(vgl. Sw-Paket, -Bibliothek, -Entwicklungsumgebung, OpenGL, frühe Windows-Versionen)
- **Systemprogrammierung** ist allgemein die Erstellung von Programmen, die den Ablauf 'systeminterner' Vorgänge regeln.
- Ein **Programm** ist die Planung einer Reihe von Handlungen (einschließlich Teilhandlungen und Details), die auf ein einheitliches Ziel hinsteuern.

- Definition: Als **System** bezeichnen wir
 - Eine Menge von **Komponenten**
(Gegenständen, Individuen, Größen, Prozessen, Ideen),
 - die untereinander in einer **kausalen Wechselwirkung** stehen
 - und von ihrer **Umgebung** entweder als abgeschlossen oder als in einer wohldefinierten **Beziehung** stehend betrachtet werden können
 - sowie
die Gesamtheit der **unter ihnen** herrschenden **Beziehungen**.

- Im Sinne dieser Definition kann jedes System in beliebig viele **Teil- oder Subsysteme** zerlegt werden, sofern dies sinnvoll erscheint.
- Die Existenz von Systemen ist meist mit der Erfüllung eines **Zweckes** verbunden (vgl. technische, politische, soziale, Regal-, Gleichungs- oder Planetensysteme).
- Der Begriff “System” ist ein **Idealtypus**:
Komponenten sind selten abgeschlossen, nicht immer unterscheidbar, Beziehungen kaum einmal “wohldefiniert”
z.B.: Verkehrssysteme, politische Systeme –
Ggs.: philosophische, mathematische (Gleichg.-) Systeme
- **Software**-Systeme lassen sich in sehr guter Näherung durch **Idealtypen** beschreiben (trotz Digitalisierungsfehler, Fremdeinflüssen durch Betriebssystem u.ä.)

Drei Software-Schichten des Sw-Gesamtsystems:



Meistverwendete Sprache in der systemnahen Programmierung: C

Beispiel / Übung:

- Erstellen Sie ein kurzes, plattform-unabhängiges C-Programm („Konsolenanwendung“), das sich in die Reihe bereits gestarteter Kopien einordnet, sich nach 20 sec (Rechenzeit) abmeldet und sich nach weiteren 20 sec selbsttätig beendet.
(Für Hedonist/inn/en: wahlweise vorzeitige Ausführung des jeweiligen Schrittes durch <CR> und Beendigung durch <ESC> über die Funktion `_kbhit()`.) (Memo.exe)
- Benutzen Sie bei der Programmierung ausschließlich Anweisungen aus dem Sprachumfang von C (d.h. u.a.: aktives Warten). Zur Simulation eines gemeinsamen Speicherbereichs können Sie eine formatierte Datei verwenden. Sie können sich dabei am Programmbeispiel für die Zeitmessung bei R/W-Operationen orientieren.
- Bauen Sie Ihr Programm so aus, daß es die benötigte Datei im aktuellen Verzeichnis einrichtet, (erst:) wenn es die erwartete Verzeichnisstruktur ("`.. /dat /`") nicht vorfindet. (MemoTest.exe)
- Weitere Hinweise und Hilfen: s. Übungsblatt.

Beispiel / Übung:

```
/*          MemoTest.c (gekuerzt)          * / Prof. Dr. A. Christidis • WS 2014/15

int main(void)
{ int j1=0, j2=WDHLG;
  FILE *memo;
  clock_t tj1, tj2;

  printf ("Aris Christidis:\n\r");
  printf ("Formatierte Datei (fscanf/fprintf):\n\r");
  _getch(); tj1 = clock();
  if ((memo=fopen("../dat/Memo.txt","r+"))==NULL &&
      (memo=fopen("../dat/Memo.txt","w+"))==NULL)
      return(-1);

  for (j1=0; j1<WDHLG; j1++)
  { fprintf(memo,"%10d", j1); fflush(memo); rewind(memo);
    fscanf(memo,"%10d", &j2); rewind(memo);
  }
  tj2 = clock(); fclose (memo);
  printf ("Zeit fuer %d RW-Operationen: %.2lf msec\n\n\r",\
         j1, ((double)(tj2-tj1)/CLOCKS_PER_SEC)*1000);
  /*...*/ _getch(); return(0);
}
```

```
#include <stdio.h>
#include <conio.h>
#include <time.h>
#define WDHLG 10000
```

Anmerkungen zu MemoTest.c:

- C-Konvention: „Kurzschluß-Auswertung“ von `if`-Abfragen (Ggs.: „vollständige Auswertung“). Bedingungen werden nur solange ausgewertet, wie sich ihr Ergebnis ändern kann, d.h.:
 - in `||`-Verknüpfngn bis zum ersten Ausdruck mit Ergebnis `!=0`
 - in `&&`-Verknüpfngn bis zum ersten Ausdruck `=0` (bzw. `NULL`).
- Unterbrechungszeichen für Zeilenwechsel innerhalb eines Ausdrucks: „`\`“ (bei Parameter-Aufzählung unnötig)

Programme, deren Zusammenwirken ‚wohldefinierte‘ Beziehungen der Rechner-Ressourcen* untereinander und zum Anwender (bzw.: Anwendung)** herstellen, bilden das Betriebssystem.

*Komponenten; **Umgebung

Ein Betriebssystem erfüllt 2 Aufgaben (außen / innen):

- Übernahme der (schwierigen) Programmierung einzelner Hw-Komponenten und deren Zusammenspiels. Für den Anwender verhält sich das Betriebssystem wie eine **Virtuelle Maschine**.

Wichtig: Einf. von Abstraktionsstufen / **Information hiding**

- Bei mehrfacher gleichzeitiger Nutzung (Programme, Menschen): Regelung des Zugriffs auf Hw-Komponenten, Daten, Rechenzeit: **Ressourcen-Verwaltung**.

Neuer Übungsschwerpunkt gegenüber bisheriger Anwendungsprogrammierung

- Bestrebungen zur Hw-Abstraktion seit Z3 (K.Zuse, 1941)
- 1960er: Einf.v.Dialogbetrieb (vs. Stapelbetrieb / Operator)
 - Mitte der 60er: K.Thompson, D.Ritchie (Bell Labs, AT&T) und MIT-Forscher entwickeln MULTICS: Multiuser-BS f. Mainframe GE645
 - Kein Einsatz (tech.Mängel) ⇒ Weiterentwicklung durch Thompson („Space Travel“) ⇒ Verballhornung durch B.Kernighan: „UNICS“
 - 1969: Betriebssystem UNICS, ab 1970: Unix (in Assembler)
Gleichzeitig (Thompson, Ritchie): A (BCPL-basiert) ⇒ B ⇒ C.
 - 1973: Unix erstes BS größtenteils in einer Hochsprache (kaum 1000 Zeilen Maschinencode) ⇒ Portierbarkeit!
Keine Vermarktung durch AT&T wg. US-Kartell-Bestimmungen
 - Ab ca. 1975: Abgabe zum Selbstkostenpreis an Univers.: Univ. of California ⇒ Berkeley Software Distribution (BSD) ⇒ Erweiterungen
 - 1984: IEEE/POSIX ⇒ US-Standardisierungsvorgabe; 1988: ANSI-C
 - 1980-1990: Xenix (Microsoft, ab Mitte 80er: Santa Cruz Operation)
 - 1993: Windows NT; 1991: Linux 0.02 (Linus Torvalds, FIN)

Bemerkungen zu den vorausgegangenen Folien:

- **Information hiding:** Entwurfsprinzip, bei dem Module (=in sich geschlossene Software-Komponenten) als Black Boxes aufgebaut werden und somit ihren inneren Aufbau den Anwender/inne/n gegenüber verbergen. Dies erfolgt aus Gründen der Sicherheit (Manipulation, Copyright) oder zur Wahrung der Flexibilität (Änderungen unter Beibehaltung der Schnittstellen).
- Mit Win2000 erreichte Windows den Umfang v. 30 Millionen Codezeilen – u. d. US-Fachblatt Smart Reseller zufolge mit 63.000 potenziell bekannten Bugs.
- **ANSI:** American National Standards Institute (US-Normungsinstitution)
- **ASCII:** American Standard Code for Information Interchange (7-Bit-Code, 1968)
- **AT&T:** American Telephone & Telegraph Company (gegr. 1885)
- **BCPL:** Basic Combined Programming Language, entwickelt in den 60ern an den Univ. von London u. Cambridge (GB)
- **Lisa:** Local Integrated Software Architecture
- **MIT:** Massachusetts Institute of Technology, Boston
- **MS-DOS:** Microsoft Disk Operation System
- **MULTICS:** MULTiplexed Information and Computing System
- **UNICS:** UNiplexed Information and Computing System. Der Scherz bestand darin, daß, während Multics mehrere Alternativen zur Erlangung seines Ziels untersuchte, Unics sich nur auf eine einfache konzentrierte.
- **Xerox PARC:** Palo Alto Research Center der Xerox Corporation

- Heute: ca. ein Dutzend Betriebssysteme; Marktführer: MS
- Gründe für d. PC-Erfolg (Debüt: IBM 1981/ mit MS-DOS)
 - Mächtiger Anbieter (IBM):
Gewähr für Fortbestand & Anbindung an Großrechner
 - Modulares Bau-Konzept:
Nach Bedarf aufrüstbar & Günstige Nachbauten Dritter
 - Orientierung am Massenmarkt („Schneeball-Effekt“):
Erschwinglicher Preis & Kurze Einarbeitung
- Paradigmenwechsel im Anspruch an Rechner:
≤80er: Erzeugung v. Ergebnissen; Darstellung auf Drucker
heute: Ständiger Dialog; Ergebnisse in div. Formen/Medien
- Wichtig für die Realisierung: Forschung am Xerox PARC
Mitte 70er: Konzept für graf. Benutzungsschnittstelle (GUI)
Apple: 1983 „Lisa“, '84 Macintosh; MS:1985 Windows 1.01