

Vorlesung

Systemprogrammierung I

Wintersemester 2000/2001

Systemprogrammierung I

© 1997-2001, Franz J. Hauck, Inf 4, Univ. Erlangen-Nürnberg [A-Org.fm, 2000-10-19 08.15]
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

- 1

A Organisatorisches

Systemprogrammierung I

© 1997-2001, Franz J. Hauck, Inf 4, Univ. Erlangen-Nürnberg [A-Org.fm, 2000-10-19 08.15]
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

A - 1

1 Dozent

- Dr.-Ing. Franz J. Hauck
 - ◆ Informatik 4, Lehrstuhl für Betriebssysteme (Prof. Dr. F. Hofmann)
 - ◆ E-mail: hauck@informatik.uni-erlangen.de

2 Übungsbetreuung

- Dipl.-Inf. Michael Golm
 - ◆ E-mail: golm@informatik.uni-erlangen.de
- Dr.-Ing. Jürgen Kleinöder
 - ◆ E-mail: kleinoder@informatik.uni-erlangen.de
- Studentische Hilfskräfte

3 Studiengänge

- ★ Diplomstudiengang Informatik (3. Semester)
- ★ Diplomstudiengang Wirtschaftsinformatik (ab 5. Semester)
- ★ Lehramtstudium Informatik
- ★ Bachelorstudiengang Computational Engineering (3. Semester)
- ★ Bachelorstudiengang Mathematik mit Schwerpunkt Informatik (3. Semester)
- Vorlesung und Übung
 - ◆ Anrechenbare Semesterwochenstunden:
4 SWS Vorlesung, 4 SWS Übungen

4 Inhalt

- Vorlesung
 - ◆ Grundlagen der Betriebssysteme (eingeschränkt auf Monoprozessoren)
 - ◆ Konzepte moderner Betriebssysteme
 - ◆ Beispielhafte Betrachtung von UNIX, Linux, Windows, Windows NT
- Übungen
 - ◆ Umgang mit den in der Vorlesung vorgestellten Betriebssystemkonzepten
 - ◆ Betriebssystemschnittstelle des UNIX/Linux-Betriebssystems (POSIX)
 - ◆ Umgang mit sogenannten *System-Calls*
 - ◆ Praktische Arbeiten: Ausprogrammieren von Übungsaufgaben

5 Vorlesung

- Termine: **Mo. von 10.15 bis 11.45 im H7**
Do. von 16.00 bis 17.30 im H7
- Skript
 - ◆ zwei Alternativen:
 - Folien der Vorlesung werden im WWW zur Verfügung gestellt und können selbst ausgedruckt werden (Vorteil: evtl. Farbe)
 - Folien werden kopiert und vor der Vorlesung ausgegeben; Gutscheilverkauf, Kosten **10,00 DM** für knapp 600 Folien auf ca. 150 S. (Vorteil: das Ausdrucken wird nicht vergessen)
 - ◆ weitergehende Informationen zum Nachlesen findet man am Besten in der angegebenen Literatur

5 Vorlesung (2)

■ URL zur Vorlesung

- ◆ http://www4.informatik.uni-erlangen.de/Lehre/WS00/V_SP1/
- ◆ hier findet man Termine, Folien zum Ausdrucken und Zusatzinformationen

■ Literatur

- ◆ A. Silberschatz; P. B. Galvin: *Operating Systems Concepts*. 4th Edition, Addison-Wesley, 1994.
- ◆ A. S. Tanenbaum: *Modern Operating Systems*, Prentice Hall, Englewood Cliffs, NJ, 1992.
- ◆ R. W. Stevens: *Advanced Programming in the UNIX Environment*. Addison-Wesley, 1992.

5 Vorlesung (3)

■ Rückmeldungen und Fragen

- ◆ Geben Sie mir Rückmeldungen über den Stoff. Nur so kann eine gute Vorlesung entstehen.
- ◆ Stellen Sie Fragen!
- ◆ Machen Sie mich auf Fehler aufmerksam!
- ◆ Nutzen Sie außerhalb der Vorlesung die Möglichkeit, elektronische Post zu versenden: hauck@informatik.uni-erlangen.de !

6 Übungen

- Aufteilung
 - ◆ Tafelübung – 2 SWS
 - ◆ Rechnerübung – 2 SWS

- Anmeldung zur Übung und Einteilung in die Übungsgruppen
 - ◆ „login: span“ an allen CIP-Workstations der Informatik
 - ◆ Login-Freischaltung wird noch bekanntgegeben
 - ◆ Benötigte Eingaben: persönliche Daten, Matrikelnummer, Termin der gewünschten **Tafelübungen**

- Übungsbeginn ist Montag, **23. Oktober 2000** mit den ersten Tafelübungen

6.1 Tafelübungen

- Termine

Mo. 08:30-10:00 Raum 00.152	Mo. 12:15-13:45 Raum 0.031
Mo. 14:15-15:45 Raum 00.151	Mo. 16:15-17:45 Raum 00.151
Di. 12:15-13:45 Raum 0.031	Di. 12:15-13:45 Raum 2.038
Mi. 08:30-10:00 Raum 00.152	Mi. 08:30-10:00 Raum 00.151
Mi. 10:15-11:45 Raum 2.038	Mi. 12:15-13:45 Raum 00.152
Mi. 14:15-16:45 Raum 00.151	Do. 10:15-11:45 Raum 00.151
Do. 12:15-13:45 Raum 2.038	Do. 14:15-15:45 Raum 00.152

- Im Zweifelsfalls sind die Termine aus dem Anmeldeprogramms richtig

6.1 Tafelübungen (2)

- Inhalt
 - ◆ Kurzeinführung in die Programmiersprache C
 - ◆ Besprechung von Übungsaufgaben
 - ◆ Klärung offener Fragen
 - ◆ Vermittlung ergänzender Informationen zur Vorlesung

6.2 Rechnerübungen

- Termine
 - Mo. 08:30-10:00 Raum 01.155
 - Di. 16:15-17:45 Raum 01.155
 - Mi. 08:30-10:00 Raum 01.155
 - Mi. 12:30-14:00 Raum 01.155
 - Fr. 08:30-10:00 Raum 01.155
 - Fr. 12:30-14:00 Raum 01.155

6.2 Rechnerübungen (2)

■ Inhalt

- ◆ Lösung der Übungsaufgaben (durch die Studierenden)
- ◆ Raum 01.155 ist reserviert
(Vorrang am Rechner für Übungsteilnehmer)
- ◆ ein Übungsleiter steht für Fragen zur Verfügung

7 Studien- bzw. Prüfungsleistungen

■ keine

- ◆ Informatik Lehramt

■ Schein

- ◆ Informatik (Diplom)
- ◆ Mathematik mit Schwerpunkt Informatik (Bachelor)

■ studienbegleitende Prüfung

- ◆ Computational Engineering (Bachelor)
- ◆ Wirtschaftsinformatik (Diplom)

7.1 Schein

- ★ Verpflichtende Abgabe von Übungsaufgaben
 - ◆ Abgabe erfolgt über ein spezielles Abgabeprogramm
 - ◆ Aufgaben können leider nicht bewertet werden
 - ◆ Aufgaben werden auf Plausibilität geprüft und auf Abschreiben getestet

- ★ Klausur am Semesterende
 - ◆ Zulassung zur Klausur nur wenn eine ausreichende Bearbeitung der Übungsaufgaben erfolgt ist (50%)
 - ◆ bei ausreichender Leistung in der Klausur wird der Schein vergeben

 - ◆ Bei guter Mitarbeit in den Übungsaufgaben ist die Klausur leicht zu bestehen.

7.2 Studienbegleitende Prüfung

- ★ Keine verpflichtende Abgabe von Übungsaufgaben
 - ◆ aber **dringend** empfohlen

- ★ Klausur am Semesterende bzw. in den Semesterferien (genauer Termin muss noch abgestimmt werden)
 - ◆ für die Note in der Klausur werden entsprechend Leistungspunkte bzw. Credit Points vergeben

 - ◆ Klausur enthält neben dem Stoff der Übungsaufgaben auch Stoff der Vorlesung, vor allem solchen, der mit den Übungsaufgaben in Zusammenhang steht.

B Einführung

1 Warum Systemprogrammierung I?

- Rasche Einarbeitung in spezielle Systeme
 - ◆ MVS, BS2000, VM, Solaris, Unix, Windows NT, Windows 95/98, MS/DOS
- Strukturierung komplexer Programmsysteme
 - ◆ Unterteilung in interagierende Komponenten
- Konzeption und Implementierung spezialisierter Systeme
 - ◆ Eingebettete Systeme (*Embedded Systems*)
 - ◆ Automatisierungssysteme
- Erstellung fehlertoleranter Systeme
- Verständnis für Abläufe im Betriebssystem
 - ◆ Ökonomische Nutzung der Hardware
 - ◆ Laufzeitoptimierung anspruchsvoller Anwendungen

1.1 Phänomene der Speicherverwaltung

■ Beispiel: Initialisierung von großen Matrizen

◆ Variante 1:

```
#define DIM 6000

int main()
{
    register long i, j;
    static long matrix[DIM][DIM];

    for( i= 0; i< DIM; i++ )
        for( j= 0; j< DIM; j++ )
            matrix[i][j]= 1;

    exit(0);
}
```

1.1 Phänomene der Speicherverwaltung (2)

■ Beispiel: Initialisierung von großen Matrizen

◆ Variante 2:

```
#define DIM 6000

int main()
{
    register long i, j;
    static long matrix[DIM][DIM];

    for( j= 0; j< DIM; j++ )
        for( i= 0; i< DIM; i++ )
            matrix[i][j]= 1;

    exit(0);
}
```

◆ Schleifen sind vertauscht

1.1 Phänomene der Speicherverwaltung (3)

■ Messergebnisse (Sun UltraSparc 1, 128M Hauptspeicher)

◆ Variante 1:

User time= 3,69 sec; System time= 1,43 sec; Gesamtzeit= 22,03 sec

◆ Variante 2:

User time= 21,86 sec; System time= 2,33 sec; Gesamtzeit= 86,39 sec

■ Ursachen

◆ Variante 1 geht sequentiell durch den Speicher

◆ Variante 2 greift versetzt ständig auf den gesamten Speicherbereich zu

Beispiel: `matrix[4][4]` und die ersten fünf Zugriffe

Variante 1

1	2	3	4	5															
---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Variante 2

1	5			2				3					4						
---	---	--	--	---	--	--	--	---	--	--	--	--	---	--	--	--	--	--	--

1.1 Phänomene der Speicherverwaltung (4)

■ Ursachen

◆ Logischer Adressraum

- Benutzte Adressen sind nicht die physikalischen Adressen
- Abbildung wird durch Hardware auf Seitenbasis vorgenommen (Seitenadressierung)
- Variante 2 hat weniger Lokalität, d.h. benötigt häufig wechselnde Abbildungen

◆ Virtueller Speicher

- Möglicher Adressraum ist größer als physikalischer Speicher
- Auf Seitenbasis werden Teile des benötigten Speichers ein- und ausgelagert
- bei Variante 2 muss viel mehr Speicher ein- und ausgelagert werden

1.2 Phänomene des Dateisystems

- Beispiel: Sequentielles Schreiben mit unterschiedlicher Pufferlänge

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define BUFLLEN 8191

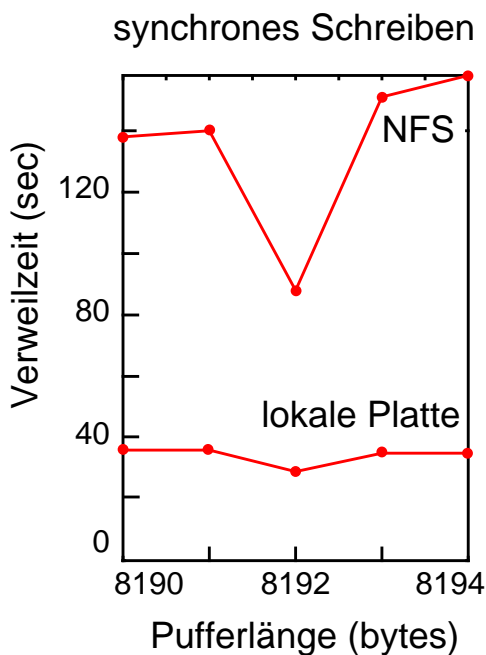
int main()
{
    static char buffer[BUFLLEN];
    int i, fd= open( "filename",
                    O_CREAT|O_TRUNC|O_WRONLY|O_SYNC,
                    S_IRUSR|S_IWUSR );

    for( i= 0; i < 1000; i++ )
        write( fd, buffer, BUFLLEN );

    exit(0);
}
```

1.2 Phänomene des Dateisystems (2)

- Messergebnisse



1.2 Phänomene des Dateisystems (3)

■ Ursachen

- ◆ Synchrones Schreiben erfordert sofortiges Rausschreiben der Daten auf Platte (nötig beispielsweise, wenn hohe Fehlertoleranz gefordert wird – Platte ist immer auf dem neuesten Stand)
- ◆ 8192 ist ein Vielfaches der Blockgröße der Plattenblocks
- ◆ kleine Abweichungen von der Blockgröße erfordern bereits zusätzliche Blocktransfers

2 Überblick über die Vorlesung

★ Inhaltsübersicht

- A. Organisation
- B. Einführung
- C. Dateisysteme
- D. Prozesse und Nebenläufigkeit
- E. Speicherverwaltung
- F. Implementierung von Dateien
- G. Ein-, Ausgabe
- H. Verklemmungen
- I. Datensicherheit und Zugriffsschutz

3 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.**“

■ 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].“

3 Was sind Betriebssysteme? (2)

■ Silberschatz/Galvin

- ◆ „... ein Programm, das als Vermittler zwischen Rechnernutzer und Rechner-Hardware fungiert. Der Sinn des Betriebssystems ist eine Umgebung bereitzustellen, in der Benutzer bequem und effizient Programme ausführen können.“

■ Brinch Hansen

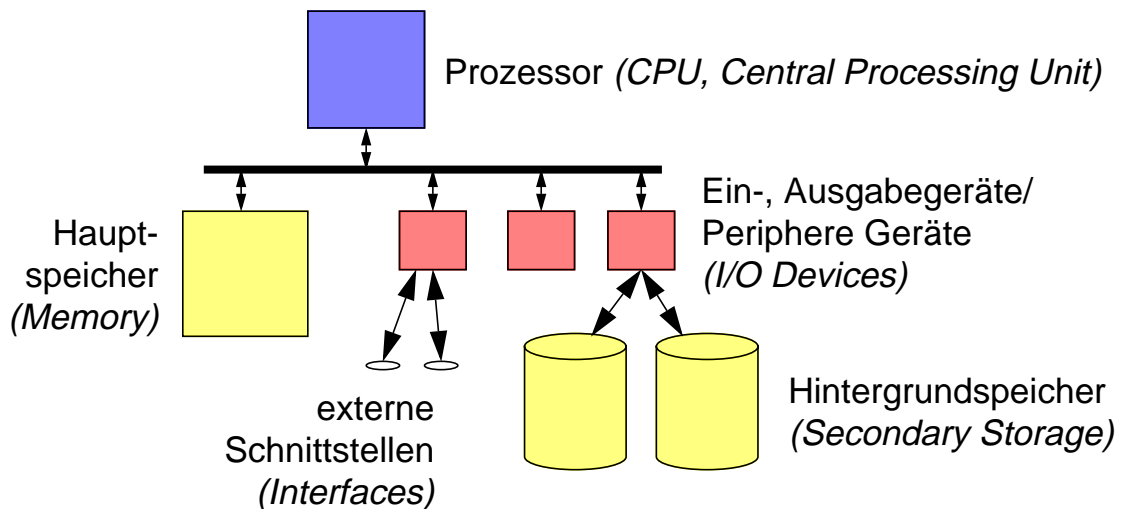
- ◆ „... der Zweck eines Betriebssystems [liegt] in der Verteilung von Betriebsmitteln auf sich bewerbende Benutzer.“

★ Zusammenfassung

- ◆ Software zur Betriebsmittelverwaltung
- ◆ Bereitstellung von Grundkonzepten zur statischen und dynamischen Strukturierung von Programmsystemen

3.1 Verwaltung von Betriebsmitteln

■ Betriebsmittel



3.1 Verwaltung von Betriebsmitteln (2)

■ Resultierende Aufgaben

- ◆ Multiplexen von Betriebsmitteln für mehrere Benutzer bzw. Anwendungen
- ◆ Schaffung von Schutzumgebungen

■ 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

3.2 Schnittstellen

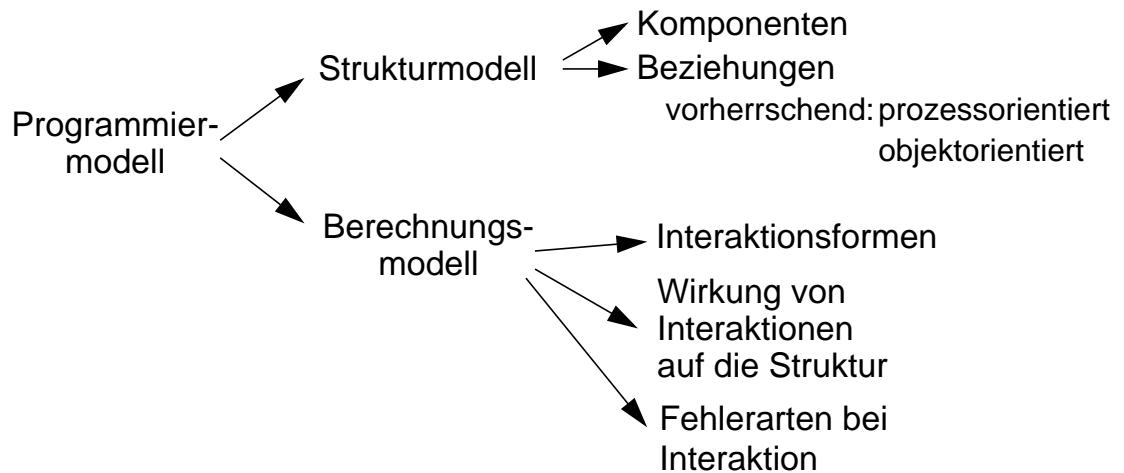
- Betriebssystem soll Benutzervorstellungen auf die Maschinengegebenheiten abbilden
 - ◆ Bereitstellung geeigneter Abstraktionen und Schnittstellen für
 - Benutzer:**
Dialogbetrieb, graphische Benutzeroberflächen
 - Anwendungsprogrammierer:**
Programmiersprachen, Modularisierungshilfen, Interaktionsmodelle (Programmiermodell)
 - Systemprogrammierer:**
Werkzeuge zur Wartung und Pflege
 - Operateure:**
Werkzeuge zur Gerätebedienung und Anpassung von Systemstrategien

3.2 Schnittstellen (2)

- Administratoren:**
Werkzeuge zur Benutzerverwaltung, langfristige Systemsteuerung
- Programme:**
„*Supervisor Calls (SVC)*“,
„*Application Programmer Interface (API)*“
- Hardware:**
Gerätetreiber

3.3 Programmiermodelle

- Betriebssystem realisiert ein Programmiermodell
 - ◆ Keine Notwendigkeit genauer Kenntnisse über Hardwareeigenschaften und spezielle Systemsoftwarekomponenten
 - ◆ Schaffung einer begrifflichen Basis zur Strukturierung von Programmsystemen und ihrer Ablaufsteuerung



3.3 Programmiermodelle (2)

- Beispiele für Strukturkomponenten
 - ◆ Dateien (Behälter zur langfristigen Speicherung von Daten)
 - ◆ Prozesse (in Ausführung befindliche Programme)
 - ◆ Klassen (Vorlagen zur Bildung von Instanzen)
 - ◆ Instanzen/Objekte
 - ◆ Prozeduren
 - ◆ Sockets (Kommunikationsendpunkte, „Kommunikationssteckdosen“)
 - ◆ Pipes (Nachrichtenkanäle)
- Beispiele für Beziehungen
 - ◆ A kann B referenzieren, beauftragen, aufrufen, modifizieren
 - ◆ Pipe P verbindet A und B

3.3 Programmiermodelle (3)

- Beispiele für Interaktionsformen
 - ◆ Prozedur-(Methoden-)Aufruf
 - ◆ Nachrichtenaustausch
 - ◆ Gemeinsame Speichernutzung
- Wirkung von Interaktionen auf die Struktur
 - ◆ Erzeugung und Tilgung von Prozessen
 - ◆ Instanziierung von Objekten
- Fehlerarten bei Interaktion
 - ◆ Verlust, Wiederholung oder Verspätung von Nachrichten
 - ◆ Abbruch aufgerufener Methoden, Ausnahmebehandlung

3.4 Ablaufmodelle

- Betriebssystem realisiert eine Ablaufumgebung
- Bereitstellung von Hilfsmitteln zur Bearbeitung von Benutzerprogrammen und zur Steuerung ihrer Abläufe.
 - ◆ Laden und Starten von Programmen
 - ◆ Überwachung des Programmablaufs
 - ◆ Beenden und Eliminieren von Programmen
 - ◆ Abrechnung (*Accounting*)