Themenfelder

SoS$_i$ ☞ System { programmierung / programme / aufrufe / funktionen } — programmierung in C; programme, aufrufe, funktionen } UNIX-ähnlicher **Betriebssysteme**

SoS$_{ii}$ ☞ Datenbanksysteme

SoS$_{iii}$ ☞ Softwaretechnik

Softwaremaschine „Betriebssystem“

- Rückgrat jedes Rechnersystems, egal ob groß oder klein, ist ein Betriebssystem
  - was Betriebssysteme sind und was nicht, ruft „Glaubenskriege“ hervor
  - entscheidend ist, dass Betriebssysteme nie dem Selbstzweck dienen
- ein Verständnis für Betriebssystemabläufe hilft, Phänomene zu begreifen
  - Eigenschaften (*features*) sind so von Fehlern (*bugs*) unterscheidbar
    - ☞ um Systemfehler kann verschiedentlich „herum programmiert“ werden
    - ☞ Systemeigenschaften können Anwendungen auch <u>un</u>möglich machen
  - das Systemverhalten kann sich positiv/negativ „nach oben“ auswirken
- Betriebssysteme sind mehr oder weniger umfangreiche *Softwaresysteme*

Phänomen — Speicherverwaltung (1)

Sei `N` „beliebig große“ natürliche Zahl und gelte: `typedef int Matrix [N][N];`

```
void preset (Matrix m, int v) {
    unsigned int i, j;
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            m[i][j] = v;
}
```

```
void preset (Matrix m, int v) {
    unsigned int i, j;
    for (j = 0; j < N; j++)
        for (i = 0; i < N; i++)
            m[i][j] = v;
}
```

☞ Welcher Unterschied besteht zwischen den beiden Implementierungen?

☞ Wirkt sich der Unterschied auf das Laufzeitverhalten aus? Weshalb ist das so?

Phänomen — Speicherverwaltung (2)

Welcher Unterschied besteht zwischen beiden Implementierungen? *Die Schleifen sind vertauscht!*

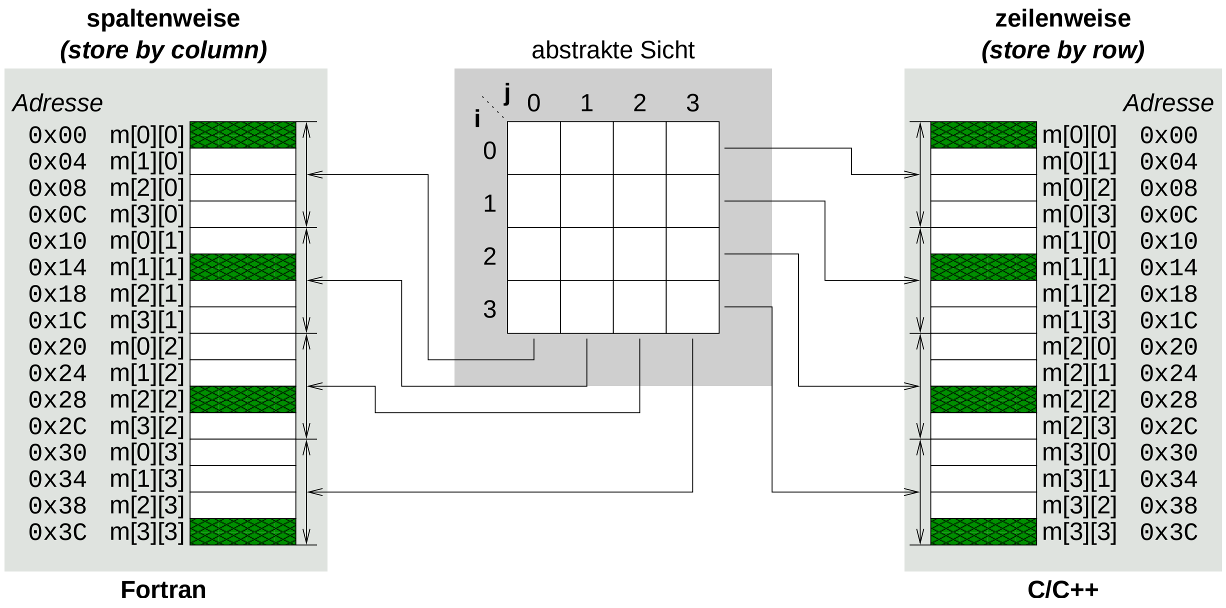
- im linken Fall wächst `j` schneller als `i`, im rechten Fall ist das umgekehrt

Wirkt sich der Unterschied auf das Laufzeitverhalten aus? *Ja und nein!*

- der **Kompilierer** bildet das 2-dimensionale Feld „`Matrix`“ ab auf einen 1-dimensionalen logischen (abstrakten) Adressraum
- das **Betriebssystem** bildet den logischen Adressraum ab auf den wirklichen, physikalischen (konkreten) Adressraum

Weshalb ist das so? *Es können sich Laufzeitunterschiede ergeben, falls das Betriebssystem die Abbildung auf Grundlage von virtuellem Speicher leistet!*

Speicherverwaltung (3)



Speicherverwaltung (4)

- der Schleifentausch führt zu unterschiedlichen Referenzfolgen auf den Speicher:

  `j` **wächst schneller als** `i` *zeilenweise* ☞ {0x00, 0x04, 0x08, 0x0C, 0x10, 0x14, 0x18, 0x1C, 0x20, 0x24, 0x28, 0x2C, 0x30, 0x34, 0x38, 0x3C}

  `i` **wächst schneller als** `j` *spaltenweise* ☞ {0x00, 0x10, 0x20, 0x30, 0x04, 0x14, 0x24, 0x34, 0x08, 0x18, 0x28, 0x38, 0x0C, 0x1C, 0x2C, 0x3C}

- spaltenweises Vorgehen „springt“ durch den Speicher, zeigt <u>keine Lokalität</u>
  - wenn das 2-dimensionale Feld (wie bei C/C++) zeilenweise gespeichert wird

- unkritisch, falls alle Speicherzugriffe gleich sind (*uniform memory access*, UMA)

Phänomen — Speicherverwaltung (5)

virtueller Speicher bedingt, dass Zugriffe auf den Arbeitsspeicher ungleichartig sein können, auch wenn die Hardware eine UMA-Architektur definiert!

- Programme sind trotz (zeitweiliger) Knappheit an Arbeitsspeicher ausführbar
  - nicht benötigte Programmteile liegen im Hintergrundspeicher (Platte)
  - sie werden erst bei Bedarf („*on demand*“) in den RAM geladen
  - das Betriebssystem implementiert den Zugriff auf die ausgelagerten Teile
  - die Einlagerung ist um Größenordnungen langsamer als RAM-Zugriffe
- als Konsequenz sind nicht alle Speicherzugriffe des Programms gleich schnell
  - je sprunghafter die Referenzfolge, desto wahrscheinlicher die Einlagerung
- Programmlokalität hat einen wesentlichen Einfluss auf das Laufzeitverhalten

Was *macht* ein Betriebssystem *aus*?

✤❉▼❒❇❉✲▲▲❙▲▼❉❍ (1)

Ewert ***et al*** Ein Computer ist, wenn er genau betrachtet wird, nur eine Ansammlung von Plastik und Metall, das zur Leitung von Strom benötigt wird. Dieser „Industriemüll“ kann somit nicht ausschließlich das sein, was wir unter einem modernen Computer verstehen, etwas, das dem Computer „Leben“ einhaucht und ihn zu dem Werkzeug unseres Jahrhunderts macht. *Es ist das Betriebssystem, das* die Kontrolle über das Plastik und Metall (Hardware) übernimmt und *anderen Softwareprogrammen* (Excel, Word, . . .) *eine standardisierte Arbeitsplattform* (Windows, Unix, OS/2) *schafft.*

✤❉▼❒❇❉✲▲▲❙▲▼❉❍ (2)

Be’triebs·sys·tem *Programmbündel, das die Bedienung eines Computers ermöglicht.*

Lexikon der Informatik *Summe derjenigen Programme, die* als residenter Teil einer EDV-Anlage *für den Betrieb der Anlage und für die Ausführung der Anwenderprogramme erforderlich ist.*

DIN 44300 Die *Programme* eines digitalen Rechensystems, *die* zusammen mit den Eigenschaften der Rechenanlage *die Grundlage der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen.*

✣❋▼❒❇❋✲▲▲❙▲▼❋❍ (3)

Hansen *Der Zweck* eines Betriebssystems [*besteht*] *in der Verteilung von Betriebsmitteln auf sich bewerbende Benutzer.*

Habermann *Eine Menge von Programmen, die die Ausführung von Benutzerprogrammen und die Benutzung von Betriebsmitteln steuern.*

Silberschatz/Galvin Ein Programm, das als *Vermittler zwischen Rechnernutzer und Rechnerhardware* fungiert. Der Sinn des Betriebssystems ist eine Umgebung bereitzustellen, in der Benutzer bequem und effizient Programme ausführen können.

Tanenbaum *Eine Softwareschicht, 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].

✣❋▼❒❇❋✲▲▲❙▲▼❋❍ (4)

Hofstadter *The operating system* is itself a programm which *has the functions of shielding the bare machine from access by users (thus protecting the system), and also of insulating the programmer from the many extremely intricate and messy problems* of reading the program, calling a translator, running the translated program, directing the output to the proper channels at the proper time, and passing control to the next user.

Kittler *Ein Betriebssystem kennt auf jeden Fall keinen Prozessor mehr, sondern ist neutral gegen ihn, und das war es vorher* [d. h., bevor die Schnittstelle zum real existierenden Prozessor softwaremäßig formuliert wurde] *noch nie.* Und auf diese Weise kann man eben jeden beliebigen Prozessor auf jedem beliebigen anderen emulieren, wie das schöne Wort lautet.

Weiterer Vorlesungsverlauf

- ☞ aus der Geschichte lernen .... ✗ Kap. 3
- ☞ Betriebssysteme „von aussen“ betrachten .... ✗ Kap. 4
- ☞ einen „Hauch“ von Softwaretechnik „einatmen“ .... ✗ Kap. 5
- ☞ Betriebssysteme „innen“ in Schichten auseinander nehmen .... ✗ Kap. 6 – 12
- ☞ den Stoff rekapitulieren .... ✗ Kap. ?

Grundzüge der Lehrveranstaltung

☞ *Zusammenhänge* stehen im Vordergrund, Spezialitäten im Hintergrund

- Betriebssystemfunktionen werden „schichtenweise“ behandelt
- die Einordnung der Funktionen im Gesamtgefüge bildet den Leitfaden
- eine logische Struktur wird skizziert, mit mehreren Ausprägungsformen
- die Veranstaltung versteht sich als Ergänzung zum klassischen Lehrbuchstoff

☞ Ausprägungen von Betriebssystemen dürfen nicht dogmatisiert werden

- etwa: ✯❄■◆| „ist besser als“ ✸❄■❄❑◗▲ — umgekehrt dito

☞ Betriebssysteme sind immer im *Anwendungskontext* zu sehen/beurteilen

Zusammenfassung

ein **Betriebssystem** (*operating system*)

- ist eine Menge von (mehr oder weniger wohl organisierten) Programmen

  die
  - Programme, Anwendungen oder BenutzerInnen assistieren sollen
  - die Ausführung von Programmen überwachen und steuern
  - den Rechner für eine Anwendungsklasse betreiben
  - eine *abstrakte Maschine* implementieren

- hat die Aufgabe, die **Betriebsmittel** des Rechners zu verwalten

  d. h., Ressourcen
  - kontrolliert nutzen zu können
  - ggf. gerecht zu verteilen

- definiert sich nicht über die Architektur, sondern über Funktionen (Dienste)