FH | JOANNEUM
University of Applied Sciences

Systemnahe Programmierung
SS 2024

Einführung

Helmut Lindner

Systemnahe Programmierung

Was ist systemnahe Programmierung?

- Mit System ist das Betriebssystem, in unserem Fall Linux, gemeint
- Das Betriebssystem verwaltet die Hardware-Ressourcen und stellt Abstraktionen dafür bereit (Prozess, Dateisystem, ...)
- Systemnahe Software ist Software, die direkt auf die Schnittstellen des Betriebssystems zugreift.

Systemnahe Software

- Dienstprogramme (Editoren, Konverter, Statistiken...)
- Werkzeuge
- Services (Webserver, Datenbanken, ...)
- Middleware ...

Systemnahe Programmierung -> Entwicklung systemnaher Software

Systemnahe Programmierung

Betriebssystem-Schnittstelle

- Das Betriebssystem stellt eine Schnittstelle für Anwendungsprogramme zur Verfügung
- POSIX Standard - „Portable Operating System Interface for UniX“.
  „Für Unix entwickelte standardisierte Programmierschnittstelle, welche die Schnittstelle zwischen Anwendungssoftware und Betriebssystem darstellt.“[1]
- Bestandteile:
  Definition der zugrundeliegenden Basiskonzepte
  C Systemaufrufen und dazugehörige Headerdateien
  Shell und Hilfsprogramme (sh, awk, vi ...)
- In POSIX kompatiblen Systemen ist die POSIX-Schnittstelle immer Teil der C Standardbibliothek **„libc“** -> daher ist C immer die erste Sprache für systemnahe Programmierung.
- Auch praktisch alle anderen Sprach-Implementierungen verwenden letztendlich die libc.

[1] Aus Wikipedia Eintrag zu „Portable Operating System Interface“

Systemnahe Programmierung

Embedded Systems

- Sehr oft ohne Betriebssystem
- Direkte Programmierung der Hardware
- Keine einheitliche Schnittstelle zur Anwendungssoftware -> jedes System ist anders ...

Behandeln wir hier nicht!

Das Betriebssystem Linux

Hauptaufgaben

- Abstraktion
- Virtualisierung
- Geräte verwalten
- Interprozesskommunikation

Abstraktion

Das Betriebssystem abstrahiert die Hardware eines Computers und stellt High-Level Konstrukte für Anwendungsprogramme zur Verfügung.

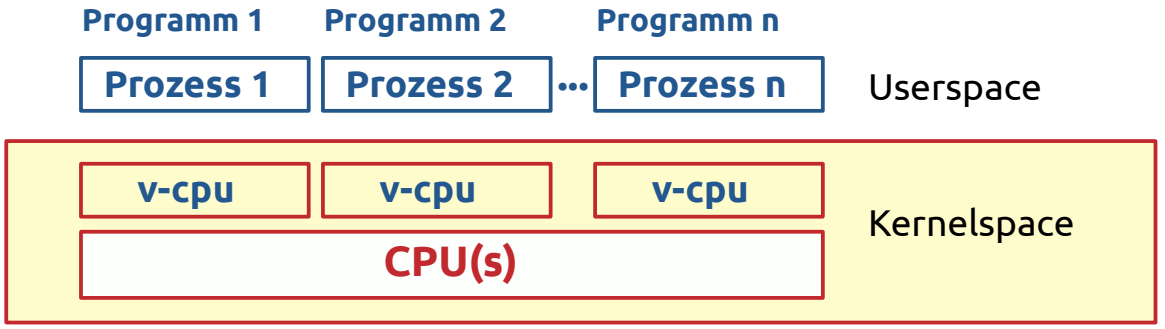
Beispiele:

- Festplatte (Zylinder, Sektoren, Blöcke) -> Filesystem
- Geräte -> Zugriff darauf in Form von Dateien (`/dev`)
- Druckerqueue
- Benutzer und Zugriffsrechte

Das Betriebssystem Linux

Prozesse - CPU Virtualisierung

Prozess - ist die Ablaufumgebung eines Programms auf einem Prozessor.
Das Betriebssystem ordnet jedem Prozess einen virtuellen Prozessor zu.



Der Betriebssystemkern (Kernel) läuft im privilegierten Modus der CPU und bildet den geschützten „Kernelspace“.

Ein Anwendungsprogramm (Prozess) läuft im User-Modus der CPU und hat keinen direkten Zugriff auf die Hardware. -> „Userspace“

Die Prozesse im Userspace sind voneinander komplett isoliert. Der Userspace kann auf die Geräte/CPU/Speicher ... nur über das Betriebssystem zugreifen.

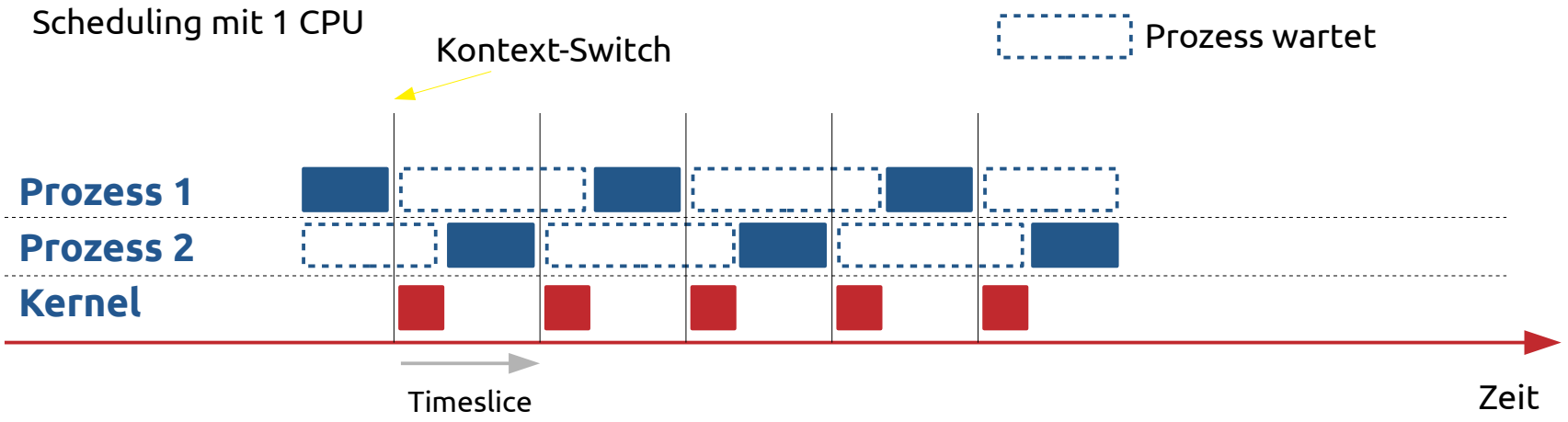
Das Betriebssystem Linux

Prozesse - Scheduling

Da im Normalfall mehr Prozesse als reale Prozessoren vorhanden sind, ordnet das Betriebssystem virtuelle Prozessoren den realen Prozessoren für eine bestimmte Dauer (Timesclice) zu.

Nach dem Ablauf einer Timeslice:

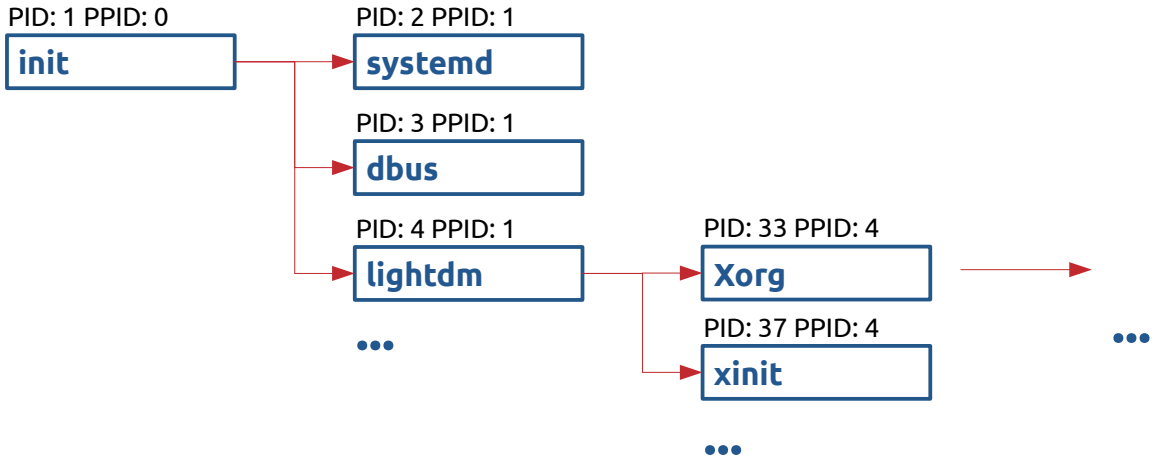
- der Kernel übernimmt die Kontrolle (Kontext-Switch)
- der aktuell ausgeführte Prozess in die Warteschlange gestellt
- der nächste ausführbare Prozess wird bestimmt (Scheduling Algorithmus)
- der nächste ausführbare Prozess wird aktiviert (Kontext-Switch)



Das Betriebssystem Linux

Prozesshierarchie

Beim Systemstart wird immer das Programm „init“ als erster Prozess gestartet. „init“ startet alle Programme deren Start für den Systemstart festgelegt ist. Diese Prozesse erzeugen weitere Prozesse - daraus ergibt sich eine Prozesshierarchie.



PID ... Process Identifier - eindeutige Nummer die jedem Prozess zugeordnet ist.
PPID ... PID des erzeugenden Prozesses (Parent-PID)

Systemnahe Programmierung

Übung 1

Öffnen sie ein Terminal und finden sie die PID des Shell-Prozesses, der gerade geöffnet ist heraus.

Verwenden sie den `ps` Befehl um folgende Fragen zu dem Prozess mit der gefundenen PID zu beantworten:

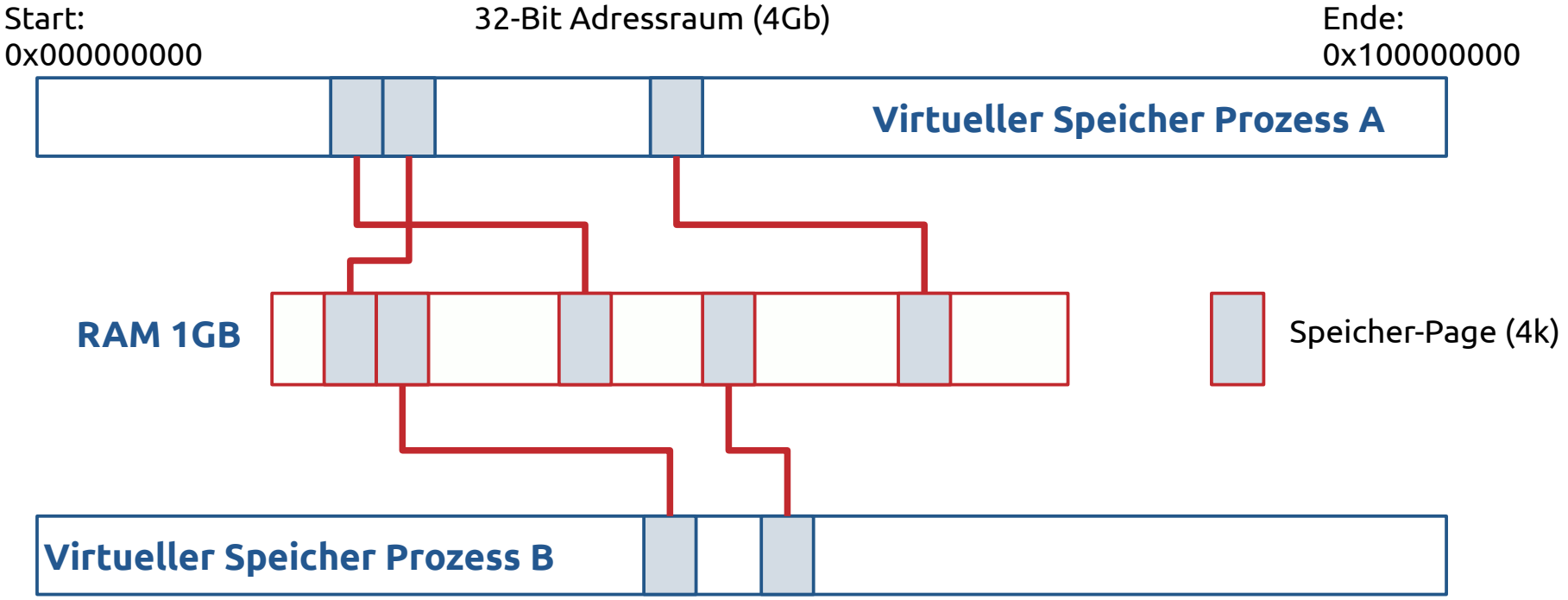
- Welches Programm wird ausgeführt?
- Welche PPID hat dieser Prozess?
- Welcher Prozess verbraucht aktuell am meisten CPU?
- Welcher Prozess hat seit dem Start am meisten CPU verbraucht?
- Welches Programm hat immer die PID 1?

Hinweise:
Der Befehl `man` zeigt die Manual-Seite eines Befehls an z.B.: `man ps`.
Die Ausgabefelder von `ps` können sie mit den „Standard Format Specifiers“ anpassen.
Die Shell-Variable `$$` enthält die aktuelle PID.

Das Betriebssystem Linux

Virtueller Adressraum

- Das Betriebssystem stellt jedem Anwendungsprozess einen eigenen virtuellen Adressraum zur Verfügung.
- Der reale Speicher wird vom Betriebssystem mit Hilfe einer MMU (Memory Management Unit) und einer Paging-Table in den virtuellen Adressraum gemappt.
- Der Speicher (virtuell und real) wird in Pages unterteilt.



Das Betriebssystem Linux

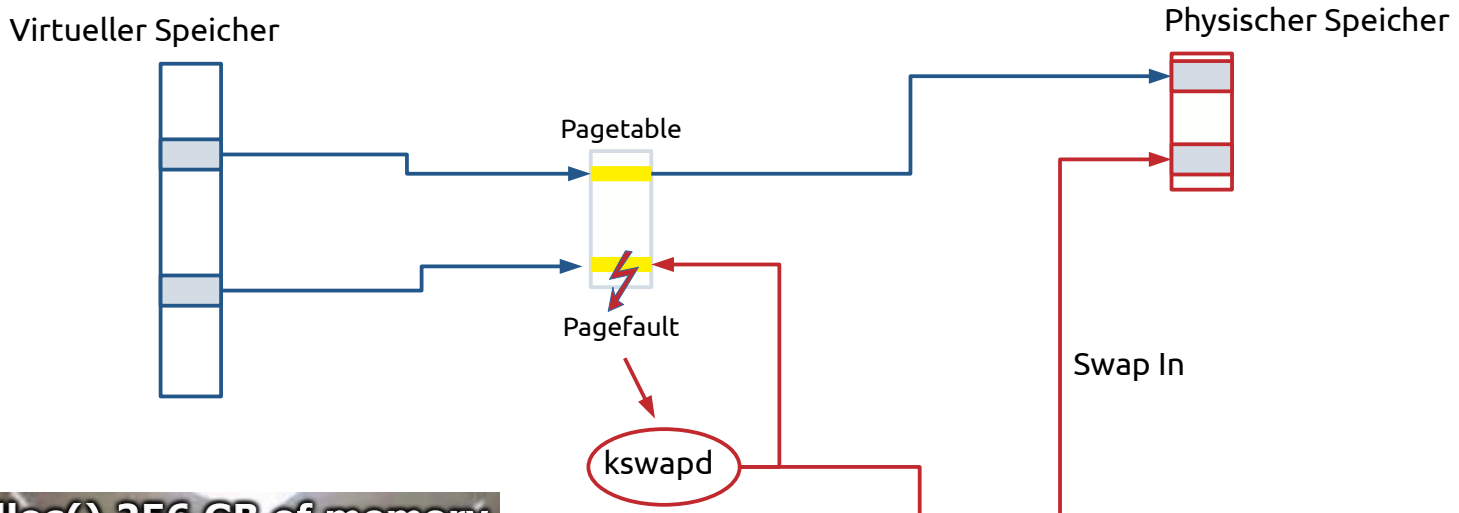
Paging/Swapping

- Paging
  Umsetzung der virtuellen Speicheradresse in ein physische Adresse.
  - Umrechnung Adresse -> Page
  - Nachschlagen der phys. Page in Pagetable
  - Berechnen der phys. Adresse

- Swapping
  Wenn nicht genügend physischer Speicher vorhanden ist, können einzelne Pages oder alle belegten Pages eines Prozesses auf ein Swap-Device ausgelagert werden -> Swap Daemon `kswapd`
  Bei Bedarf (pagefault) müssen diese wieder in den Speicher geladen werden.

Das Betriebssystem Linux

Swapping



Das Betriebssystem Linux

Geräteverwaltung

- Korrekte Initialisierung der Geräte beim Start
- Geräte werden als Files in den Userspace gemapped.
  z.B.: Festplatte /dev/sda1
- Schutz der Geräte vor fehlerhaften Anwendungsprogrammen
  Nur kontrollierter high-level Zugriff aus dem Userspace (kein direkter Zugriff auf z.B.: Statusregister ...)
- Serialisierung von parallelen Zugriffen
  Prozesse die auf Geräte zugreifen wollen, müssen warten bis andere Zugriffe abgeschlossen sind -> Warteschlange

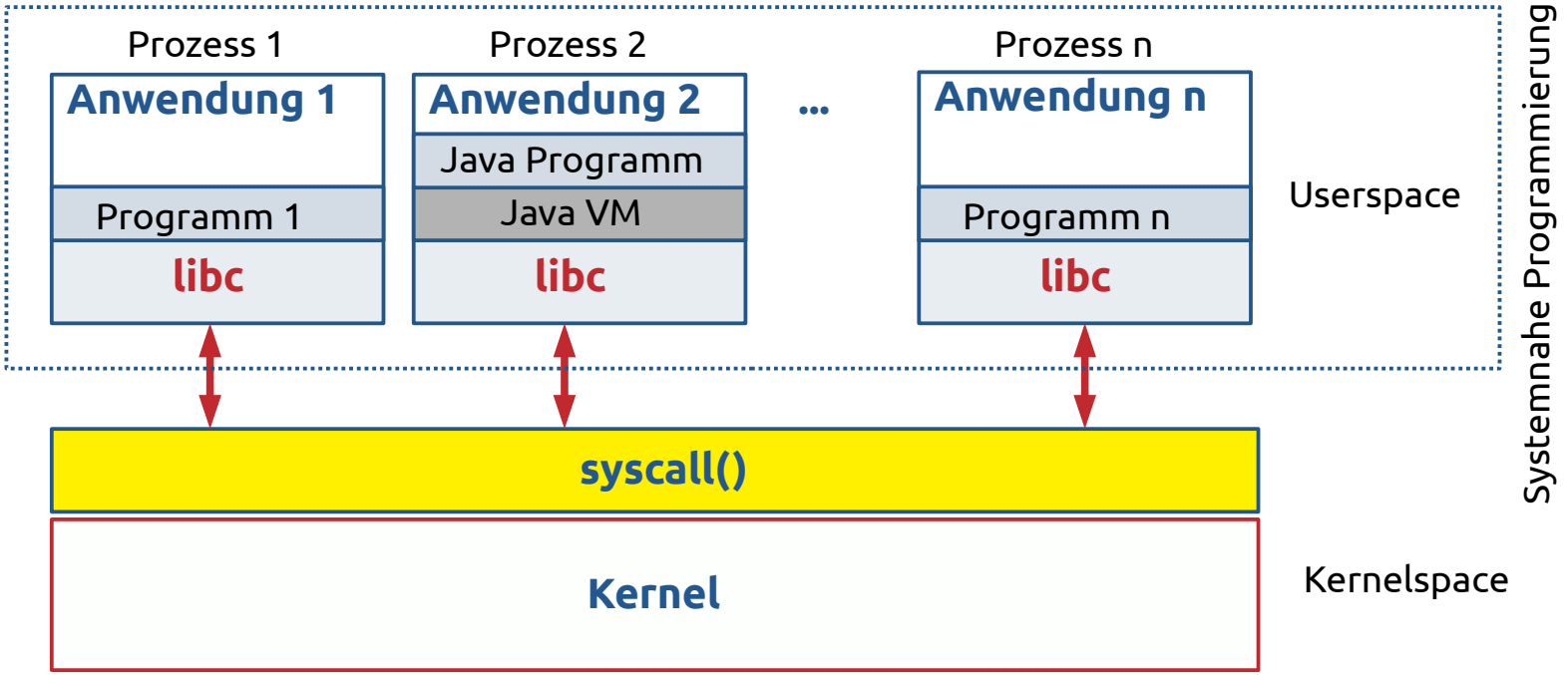
Interprozesskommunikation

- Signale
- Pipes
- Message Queue
- Shared memory

Das Betriebssystem Linux

Die Schnittstelle zum Betriebssystem

Die C Standardbibliothek **libc** stellt u.a. ein „API“ in Form von compilierten C-Funktionen zur Verfügung und mappt diese auf das Kernel-Interface



Jeder Prozess stellt der Anwendung einen komplett separierten, virtuellen Adressraum zur Verfügung. Der Zugriff auf andere Prozesse ist nicht möglich, Kernel-Zugriff ist nur über syscall() möglich.

Das Betriebssystem Linux

Übung 2

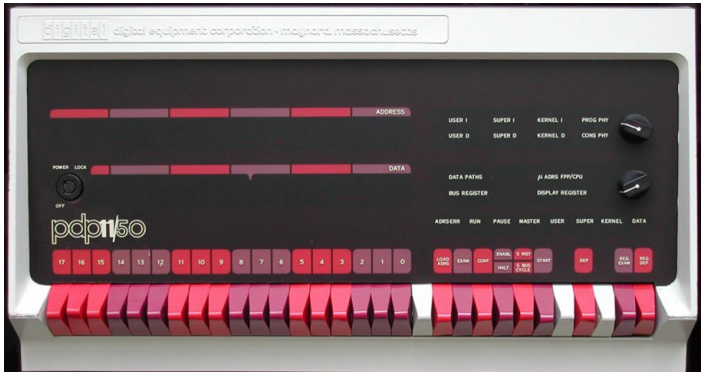
Öffnen sie ein Terminal und verwenden sie den `ps` Befehl um folgende Fragen zu beantworten:

- Wieviel physischen Speicher belegt das Programm `/usr/lib/xorg/Xorg`? Welche Format-Specifier verwenden Sie? Warum?
- Der Format-Specifier `%mem` liefert den Prozentsatz des physischen Speichers den der Prozess belegt. Verifizieren sie mit diesem Wert, ob der obige Wert stimmt.

Finden sie heraus, wo sich die Hauptkomponente C-Standardbibliothek `libc.a` im Dateisystem befindet. Interpretieren sie die Verzeichnisstruktur, was sollen die Verzeichnisnamen aussagen?

Die Programmiersprache C

Entwickelt 1972 von Dennis Ritchie - AT&T Bell Laboratories
für die Implementierung von Unix (auf PDP 11)



Weiterentwicklung

- 1972 erstmals vorgestellt
- 1978 Buch „The C Programming Language“ erschienen, erste offizielle Spezifikation der Sprache
- 1989 C89 Standard = ANSI C oder Standard C
- 1990 ANSI C wurde von ISO übernommen - C90
- 1999 C99 Standard
- 2011 C11 Standard
- 2018 C18 - nur Fehlerkorrekturen

Die Programmiersprache C

Wird heute vor allem benutzt für:

- Entwicklung von Betriebssystemen
- Systemprogrammierung
- Embedded Systems
- Performance-kritische Anwendungen und Bibliotheken

Merkmale

- Wenige Schlüsselwörter - einfache Syntax
- Strukturen
- Zeiger
- Externe Standardbibliothek - libc
- Compiliert zu Maschinencode
- Makro-Präprozessor
- Moderne Derivate: C++, C# Objective C

Die Programmiersprache C

Low-Level Sprache

- Explizite Speicherallokierung und Freigabe
- Keine Exceptions
- Keine objektorientierten Sprachkonstrukte
- Kein Range-checking
- Eingeschränkte Typ-Prüfung zur Compilezeit
- Keine Typ-Prüfung zur Laufzeit
- Kein Garbage Collector

Die Programmiersprache C

Die C-Standardbibliothek

Die C-Standardbibliothek enthält ca. 200 Funktionen aus den Bereichen

- POSIX Systemschnittstelle (Prozesse, Speicherverwaltung ...)
- Ein-/Ausgabe Funktionen
- Stringfunktionen
- Mathematische Funktionen
- Erweiterte Typfunktionen
- Funktionen für Datum und Uhrzeit
- Sprach- und Gebietsfunktionen (locale)
- Konvertierungsfunktionen
- usw.

Die C-Standardbibliothek ist standardisiert und daher auf allen Systemen gleich, d.h. die Portierung von Programmen ist vergleichsweise einfach, aber natürlich nicht so einfach wie mit Java :-(

Die C-Standardbibliothek ist im Vergleich zu modernen Umgebung, wie der Java Klassenbibliothek sehr minimalistisch.