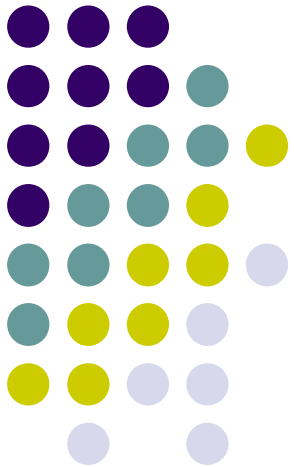
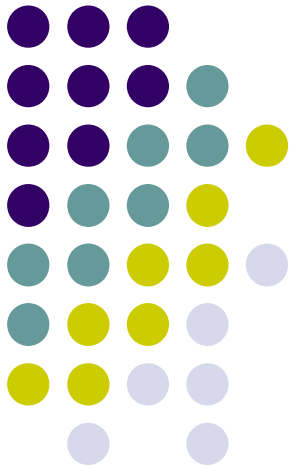
Ausgewählte Komponenten von Betriebssystemen



Lehrstuhl Prof. Paul

Ausgewählte Komponenten von Betriebssystemen

Bootstrap, Bootup & Multiboot



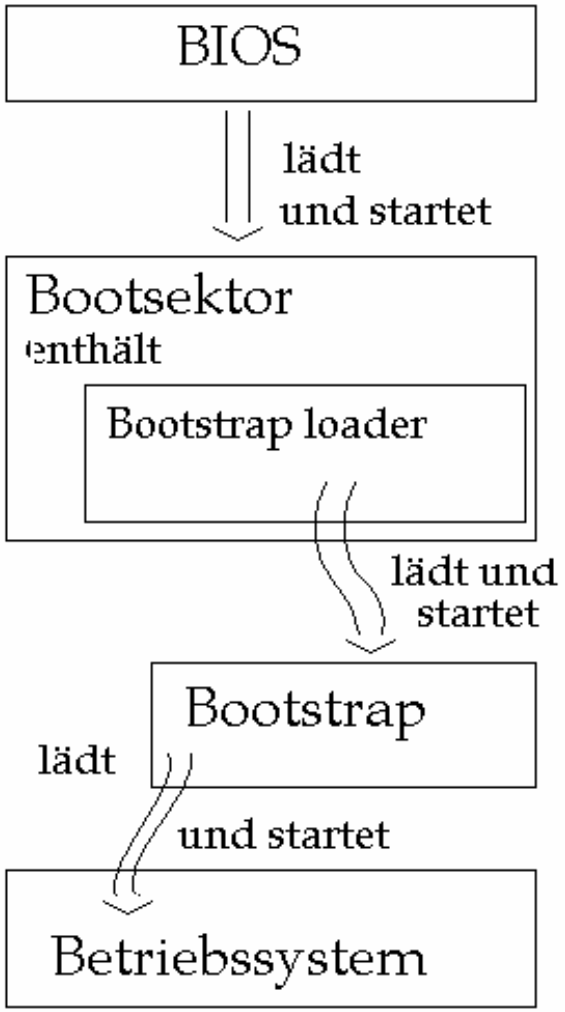
Verena Kremer
16. September 2004

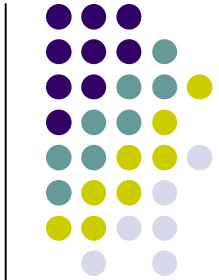
Betreuer: Sebastian Bogan

Inhalt

Bootstrap / Bootup

- Bootup
  - beginnt mit der Stromzufuhr
  - endet mit dem ersten Userprozess
  - bezeichnet den gesamten Bootvorgang
- Bootstrap
  - Teil eines Betriebssystemes
  - Lädt, entpackt und startet Systemkomponenten



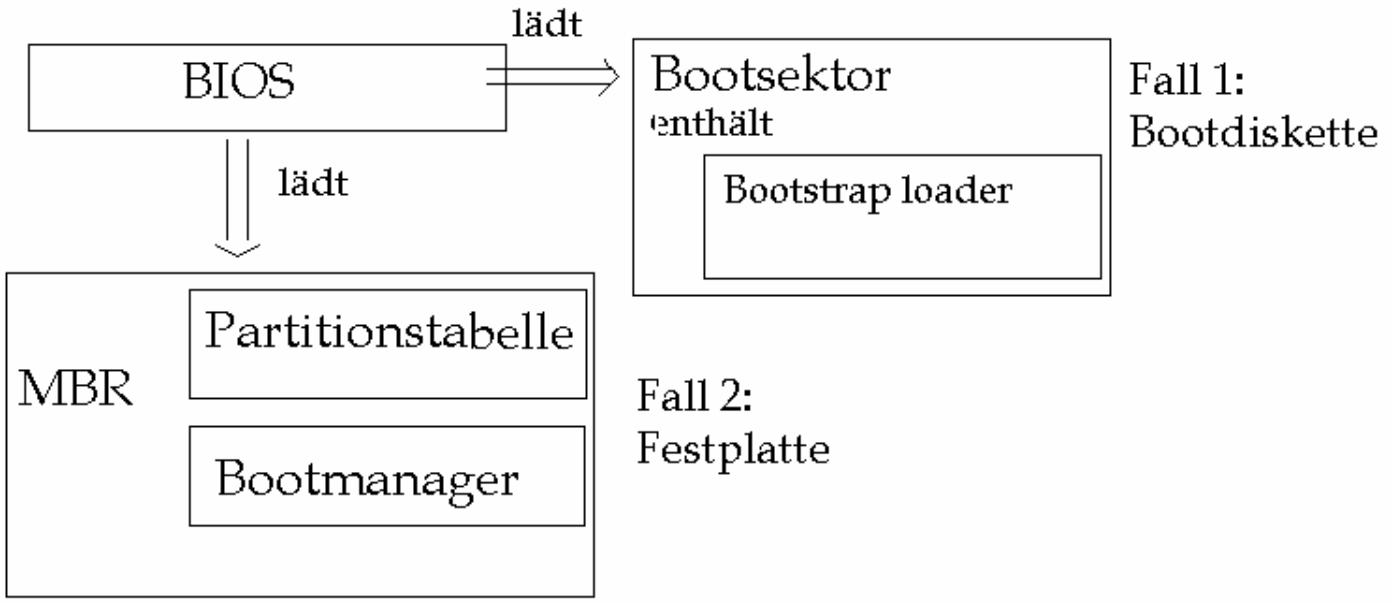


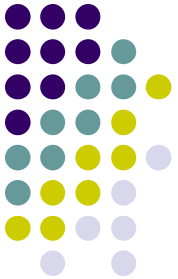
Basic Input Output System

- Steht im ROM
  - Speicher im Motherboard
  - behält Daten trotz Spannungsabfall
- Wird bei Stromzufuhr ins RAM gespiegelt und automatisch ausgeführt
- Bietet low-level Hardwareunterstützung für Betriebssysteme
- Enthält wichtigste I/O Steuerroutinen

Power-On Self-Tests

- BIOS kontrolliert Hardware
- Lädt veränderbare Daten (Uhrzeit, Bootoptionen)
- sucht nach bootfähigem Speicher
  - entweder MBR (Festplatte)
  - oder Bootsektor (Diskette)
    - gekennzeichnet durch Bootsequence / “Magic Number“





BIOS Beschränkung

- Die im BIOS enthaltenen Treiber sind minimiert
  - Es können nur die Zylinder 0 bis 1023 der Festplatte angesprochen und ausgelesen werden
  - Mehr als zwei Partitionen werden nicht erkannt
  - Sind weitere Einschränkungen vorhanden, müssen sich die aktiven Partitionen auf den „kleinsten gemeinsamen Nenner“ beschränken
- Moderne Bootmanager / Betriebssysteme verwenden eigene Treiber

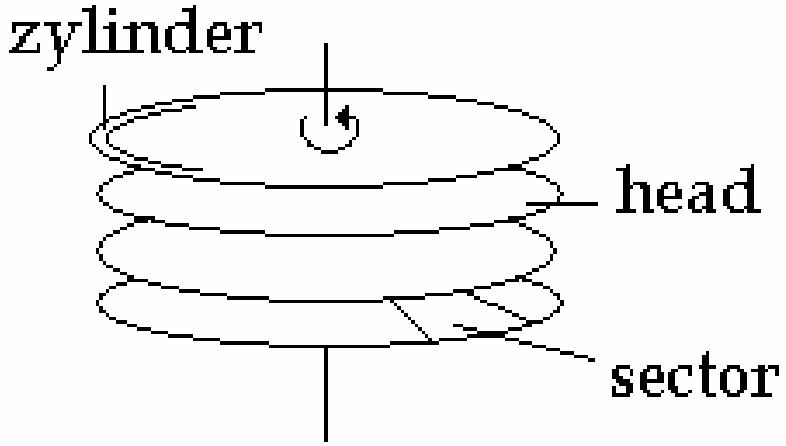
Booten über Netzwerke

- Bei manchen Betriebssystemen ist es zusätzlich möglich, einen Rechner über einen Netzwerkadapter zu booten
  - dieser Rechner (Terminal) muss keine eigene Festplatte haben – wohl aber ROM und RAM
  - beim Netzwerkbooten greift das Terminal auf eine freigegebene Partition des Servers zu und bootet von dort
  - das Terminal verwendet einen hierzu passenden Bootmanager (bsp. Netboot oder auch GRUB)
  - Bootmanager enthält alles, um Netzwerkverbindung aufzubauen, und lädt den Bootloader der Zielpartition ins RAM

Speichermedien

- Eingeteilt in Abschnitte
  - Koordinaten eines Abschnitt:

    [cylinder, head, sector]

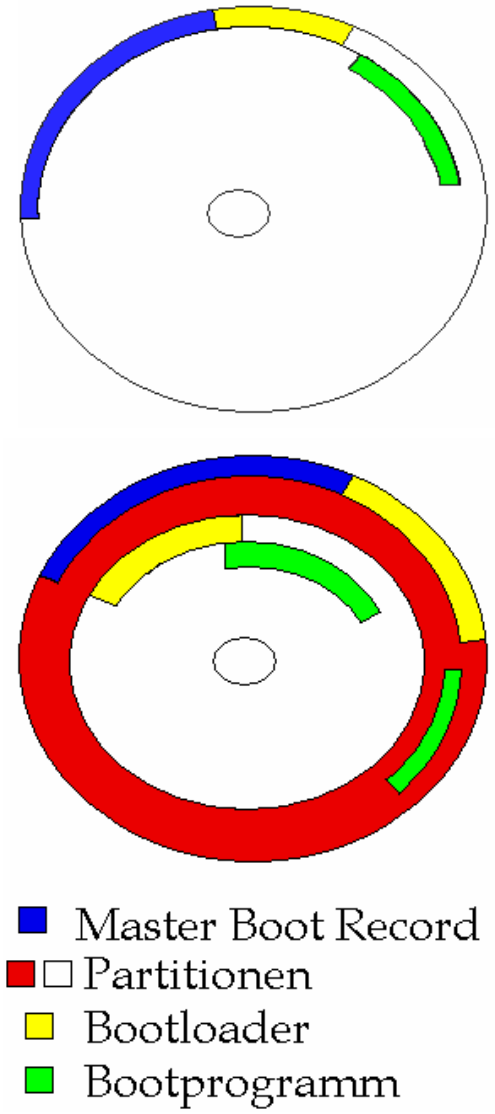


- Festplatten können mehrere (aktive) Partitionen enthalten

  => erster Teil der Festplatte (genannt MBR) enthält unter anderem Partitionstabelle

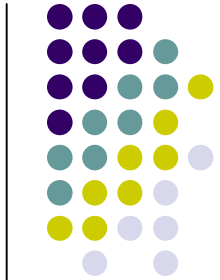
Partitionen

- Eine Partition:
  - MBR im äußersten Zylinder
  - Bootsektor am Anfang der einzigen Partition
- Mehrere Partitionen:
  - MBR im äußersten Zylinder
  - Bootsektoren am Anfang der Partitionen



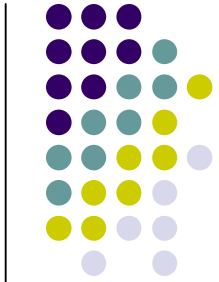
Partitionstabelle

- Steht im MBR
- Enthält je Partition
  - Boot-Flag: 1 Byte (aktiv oder nicht?)
  - Kopfnummer des Partitionsbeginns: 1 Byte
  - Sektor und Zylindernummer des Boot-Sektors: 2 Bytes
  - Systemcode: 1 Byte (Typ der Partitionen: NTFS, unformatiert…)
  - Kopfnummer des Partitionsendes: 1 Byte
  - Sektor und Zylindernummer des letzten Sektors der Partition: 2 Bytes
  - relative Sektornummer des Startsektors: 4 Bytes
  - Anzahl der Sektoren in der Partition: 4 Bytes



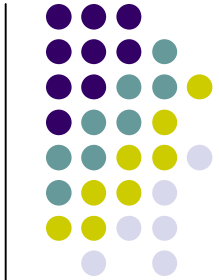
Master Boot Record

- Erster Bereich einer Festplatte
- Wird vom BIOS eingelesen:
- Enthält
  - 1. Partitionstabelle
  - 2. entweder Bootmanager oder den „Master Boot Code“ (MBC)



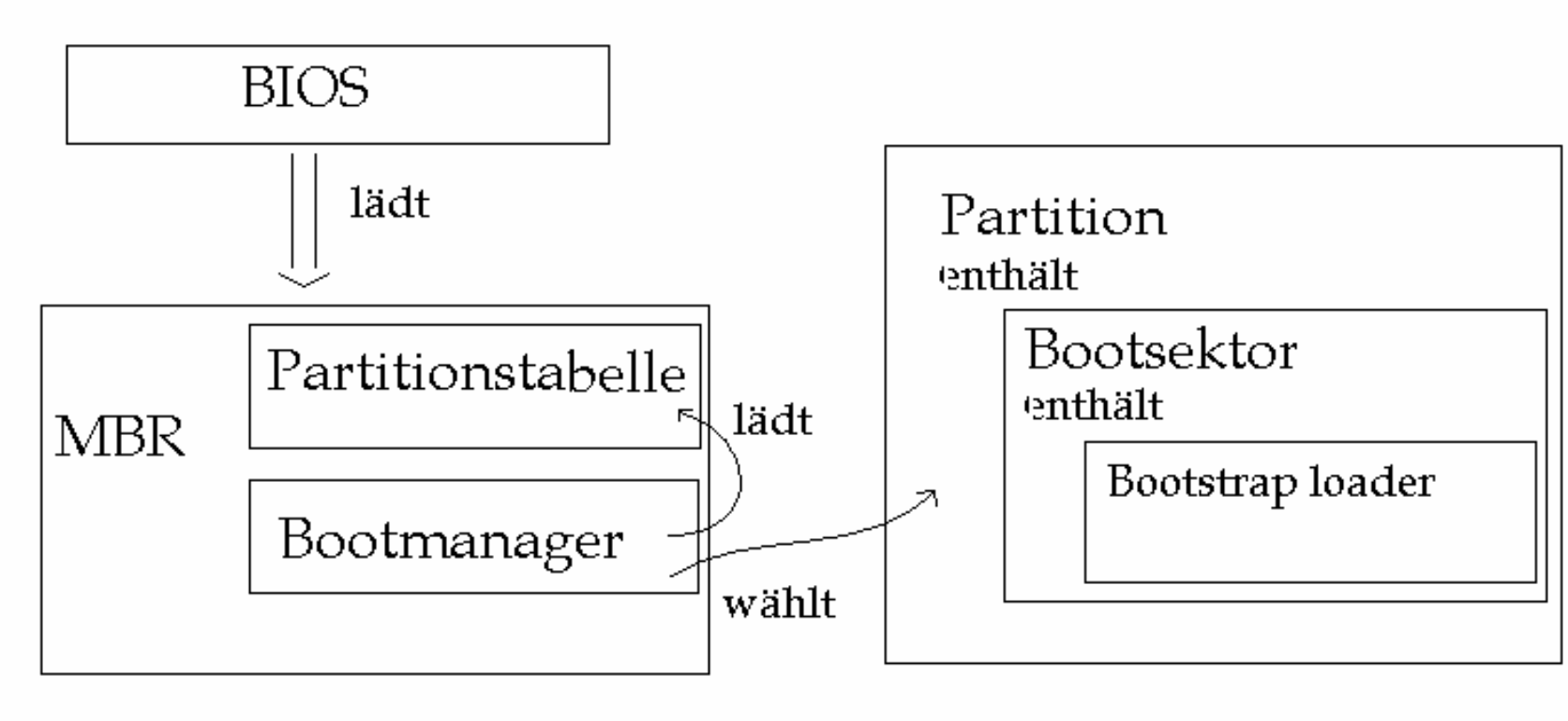
Bootmanager/ MBC

- MBC einfacher Standard-Bootmanager, der nur eine aktive Partition startet
- Bootmanager ersetzt MBC, falls mehr als eine aktive Partition zur Auswahl steht
  - liest die Partitionstabelle ein
  - stellt dem Nutzer die als aktiv markierten Partitionen zur Auswahl
  - Stellt Interface zwischen BIOS und Betriebssystem bereit
  - liest den ersten Sektor der gewählten Partition ein
  - übergibt dortigem Bootloader die Kontrolle



Multiboot

- Bootvorgang bei mehreren aktiven Partitionen mit mehreren Betriebssystemen



Beispiel GRUB

- GRand Unified Bootloader“
- Standard für Linux seit Suse 8.1

- Bringt eigene Treiber mit

  => nicht an BIOS Einschränkung gebunden

  => benötigt keine „Blocklisten“
- 2 Stages:
  - erster Teil steht im MBR (anstelle MBC)
  - zweiter Teil steht irgendwo auf Festplatte (oft als Teil von Linux)

Momentaufnahme

- entweder BIOS oder MBR haben die Kontrolle
- unser Rechner hat jetzt bootfähiges Speichermedium (Partition oder Diskette) gefunden
- der erste Sektor dieses Speichers enthält den Bootloader
- Ab hier hat Betriebssystem die Kontrolle

1. & 2. Boot(strap)loader

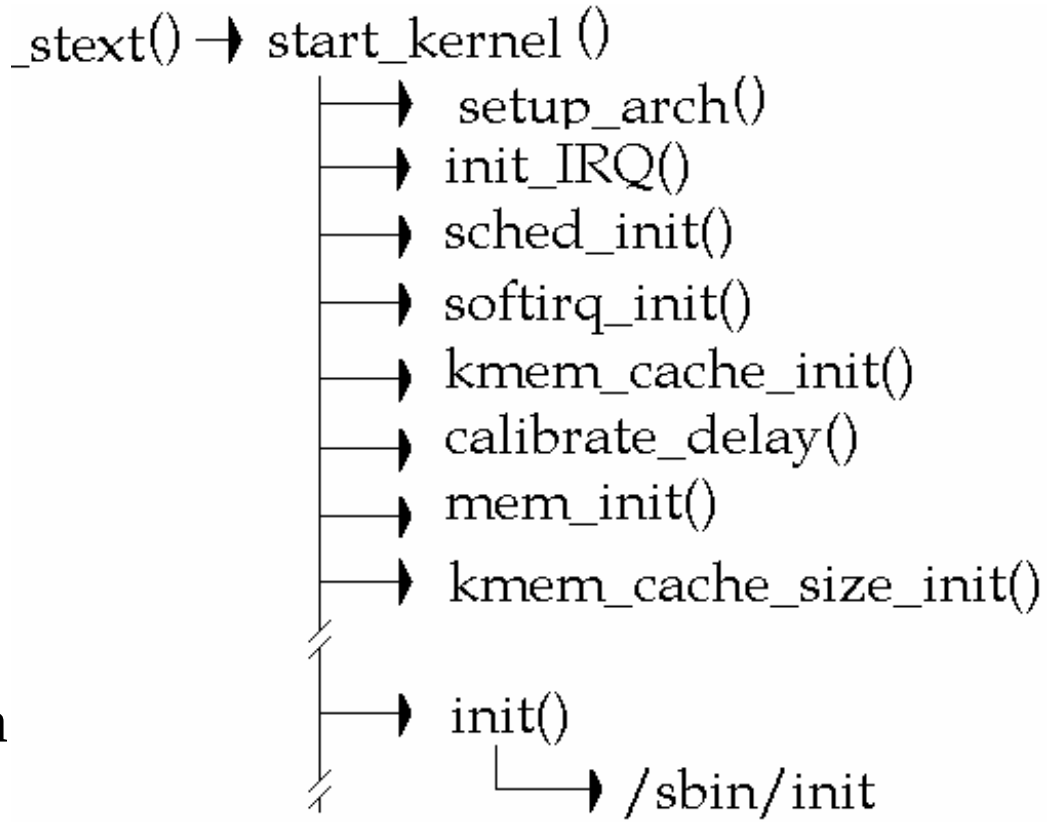
- Bootloader steht im Bootsektor (erste 510 byte)
- Lädt den Kernel in eine vorherbestimmte Position im Arbeitsspeicher
- Startet Kernel
- Manchmal unterteilt:
  - Primary Bootloader lädt eigentlichen (Secondary) Bootloader
  - auch Bootblock 1 und 2 genannt
  - genutzt, wenn Block 2 nicht in Bootsektor passt
  - typisch für DOS, NetBSD

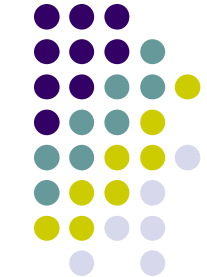
Bootstrap

- „Schnürsenkel“
- Teil des Betriebssystems
- Wird vom Boot(strap)loader geladen
- Läuft ohne externe Interrupts (Ausnahme: Reset)
- Bei Linux: Kernel
- initialisiert den Rest des Betriebssystems

Betriebssystemstart

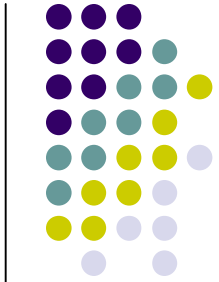
- Startvorgang:
  - entpackt Bestandteile des Betriebssystems
- Beispiel Linux:
  - Bootloader lädt nur Kernel
  - startet Kernel
  - Kernel initialisiert das restliche Betriebssystem





_stext() & start_kernel()

- Die Funktion **_stext()**
  - stellt eine primitive C Umgebung bereit
  - initialisiert Interrupt Mask, Status Register, Cache
  - lädt und startet die Funktion **start_kernel()**
- Die Funktion **start_kernel()**
  - organisiert das restliche „Hochfahren“
  - gibt das Linuxbanner aus und parst die Kommandozeile
  - Lädt und startet die übrigen Funktionen

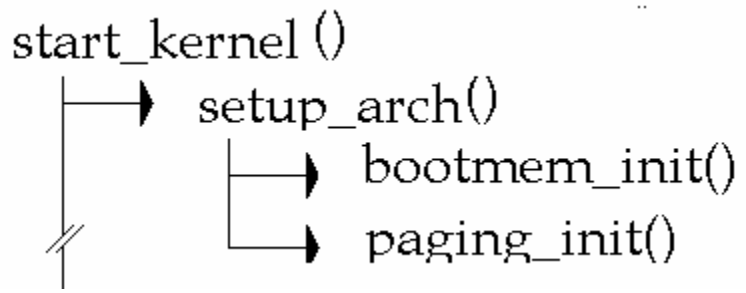


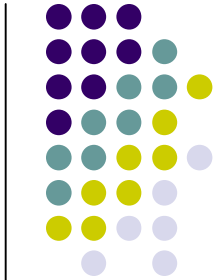
Setup_arch ()

- Die Funktion **setup_arch()**
  - Hardware- Initialisierungen
  - setzt machinevector:

    hostname+ pointer auf i/o treiber
  - sucht Größe und Standort

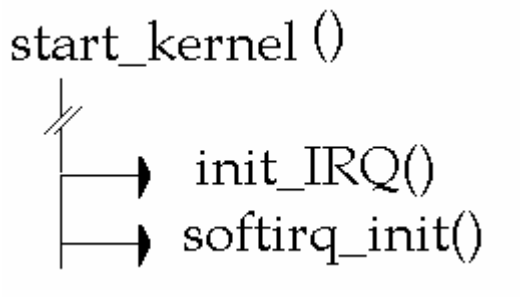
    verfügbaren Speichers
  - Startet **paging_init()** um die Memory Management Unit des hosts freizugeben.



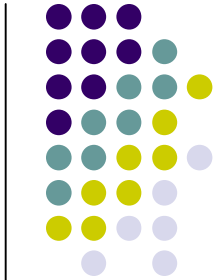


Interrupts

- Die Funktion **init_IRQ()** initialisiert die hardware-spezifischen Teile des Kernel- Interrupt- Subsystems



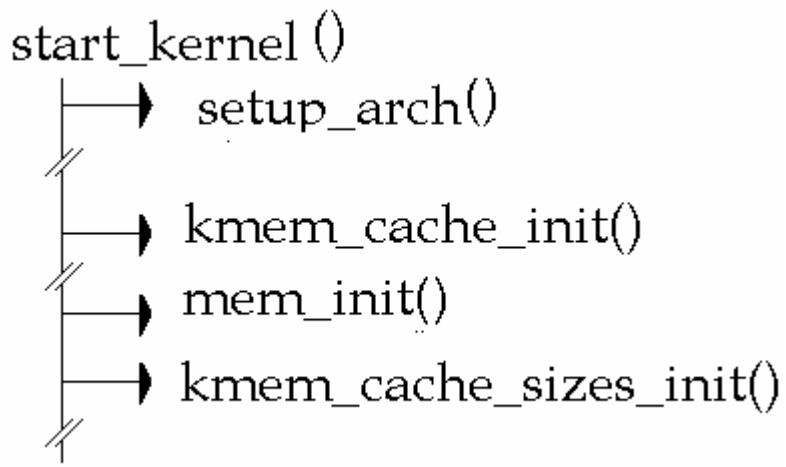
- Die Funktion **softirq_init()** startet das ***softirq*** Subsystem
  - Werden z. Bsp. zur Verbesserung des Interrupthandling bei Netzwerkprozessen benutzt



Memory Management

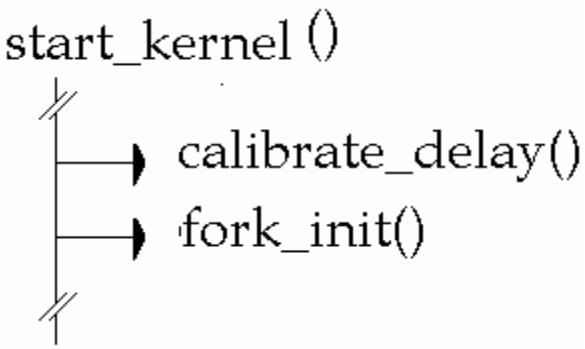
- **Kmem_cache_init()** und **mem_init()** initialisieren Memory management subsystem

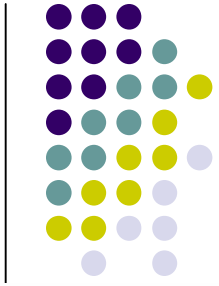
- Die Funktion **kmem_cache_sizes_init()** Beendet die Initialisierung



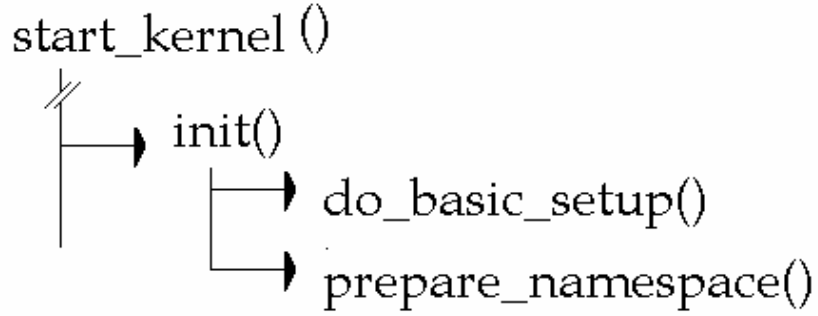
calibrate_delay()

- Die Funktion **calibrate_delay()** berechnet den „bogomip“
  - Zahl, die zur Berechnung der internen Delays verwendet wird
- liest dafür eine Zahl („jiffies“) aus BIOS aus

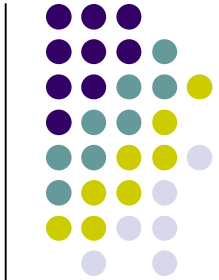




init()



- startet **do_basic_setup():**
  - initialisiert Netzwerk
  - gibt den für den Startvorgang reservierten Speicher frei
  - aktiviert Interrupts
  - Standart Input, output und errorstreams werden geöffnet
  - aktiviert Scheduling
- startet **prepare_namespace():**
  - versucht, das root- Filesystem zu mounten
  - Das root- Filesystem sind kernelintern definiert: harddisk device + Partition, die root-Verzeichnis enthält
- startet **execve(),** die wiederum die Funktion **/sbin/init** aufruft

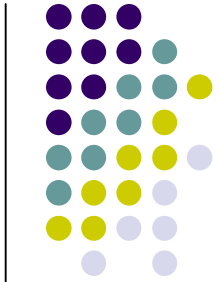


init

- /sbin/init/ ist der erste Userprozess
- Gehört nicht mehr zum Bootup

  => Bootup beendet.

Zusammenfassung



- Stromzufuhr startet BIOS
- BIOS sucht MBR / Diskette
  - evtl. wird über Bootmanager aktive Partition ausgewählt
- Bootloader auf aktiver Partition / Diskette startet und lädt Bootprogramm
- Kernel wird geladen
- Hochfahren beendet.