

# Prozesse und Threads unter Linux

SS 2008 – Proseminar: Linux für Umsteiger

Von Christopher Röcker

Ruprecht-Karls-Universität Heidelberg

Julian Kunkel, Olga Mordvinova

# Inhaltsgliederung

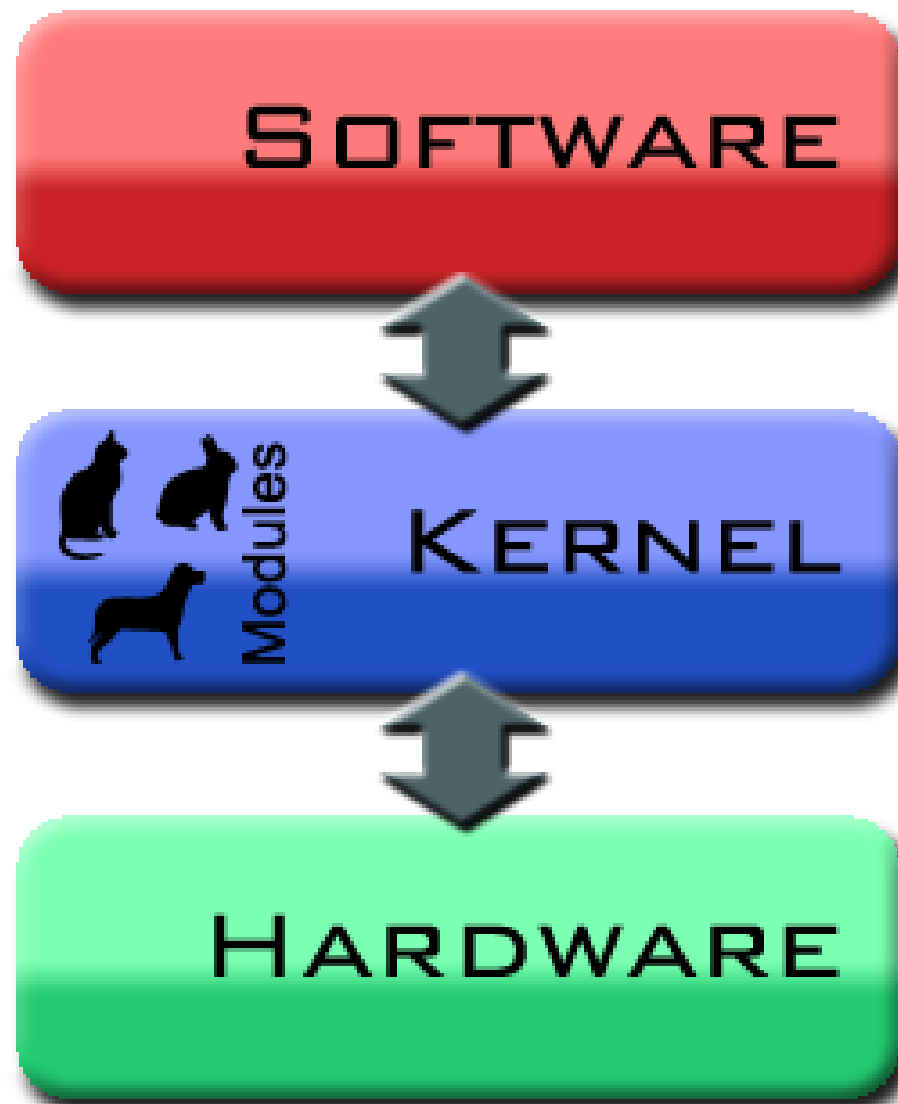
- Kernel
- Prozesse
- Threads
- Prozessverwaltung
- Interprozesskommunikation
- Der Befehl `fork()`
- `exec()`
- Wichtige Prozesse in Linux

# Kernel

- Zentraler Bestandteil eines OS (befindet sich immer im Speicher)
- Schnittstelle zwischen Hard- und Software
- Zuständig für:
  - Speicherverwaltung
  - Prozessverwaltung
  - I/O Operationen
  - ...

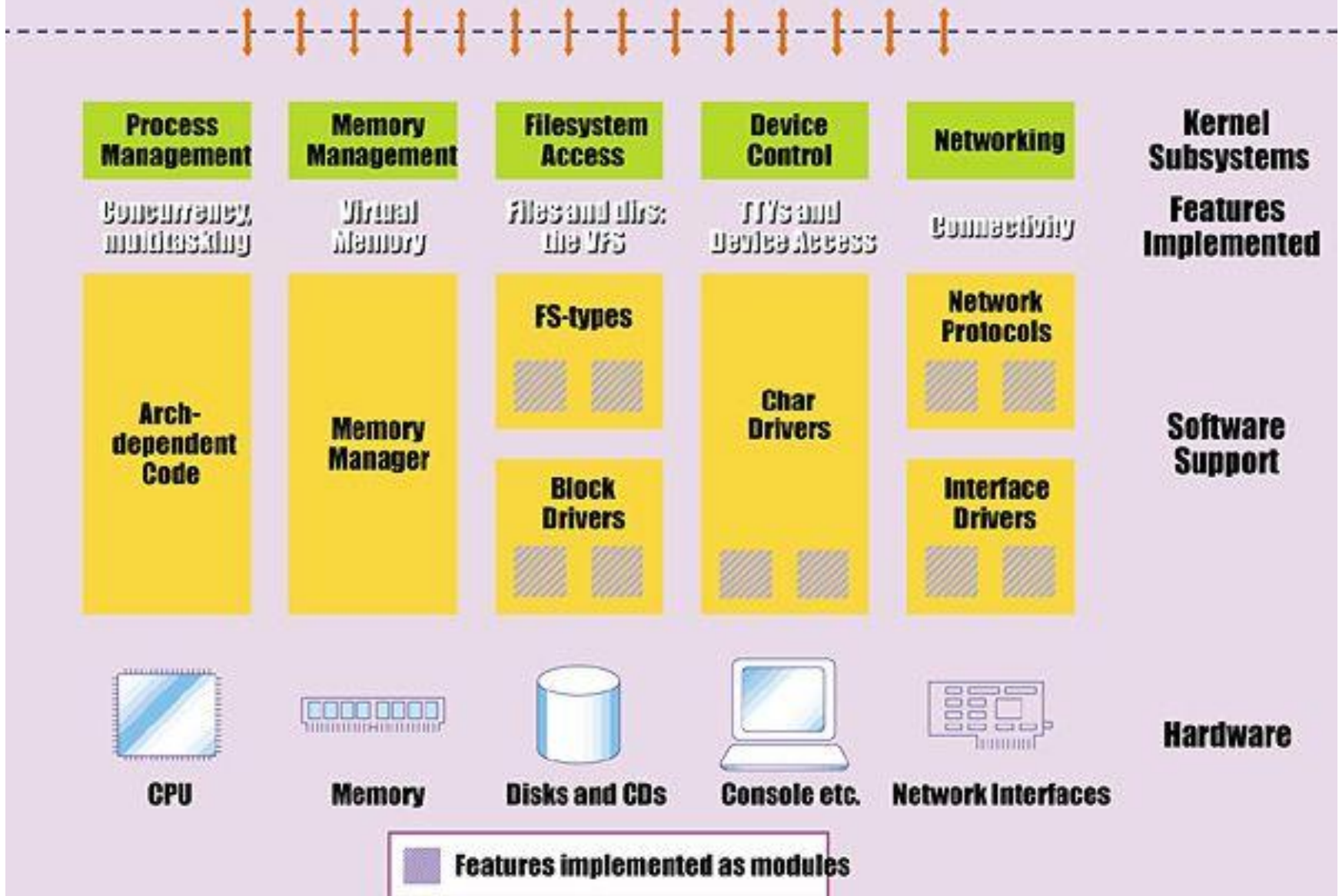
# Kernel

- Erster Linux-Kernel 1991 von Linus Torvalds
- Linux ist ein monolithischer Kernel
  - Module können geladen werden
- User-Mode (Privilegien)
- Kernel-Mode



[http://widefox.pbwiki.com/f/Monolithic\\_Kernel.png](http://widefox.pbwiki.com/f/Monolithic_Kernel.png)

# The System Call Interface



<http://www.fh-wedel.de/~si/seminare/ws01/Ausarbeitung/2.linuxtreiber/kernel.gif>

# Prozesse unter Linux

- Prozess besteht aus drei Segmenten:
  - Textsegment (auch Codesegment genannt)
  - Data-Segment (Daten des Threads)
  - Stacksegment (Stack des Threads)
- Prozesse bestehen aus Threads
- Benutzermodus und Systemmodus

# Prozesse unter Linux

- Verschiedene Prozesszustände
  - Dead
  - Ready
  - Running
  - Sleep
  - Wait
  - Zombie/Waise

# Zombies und Waisen

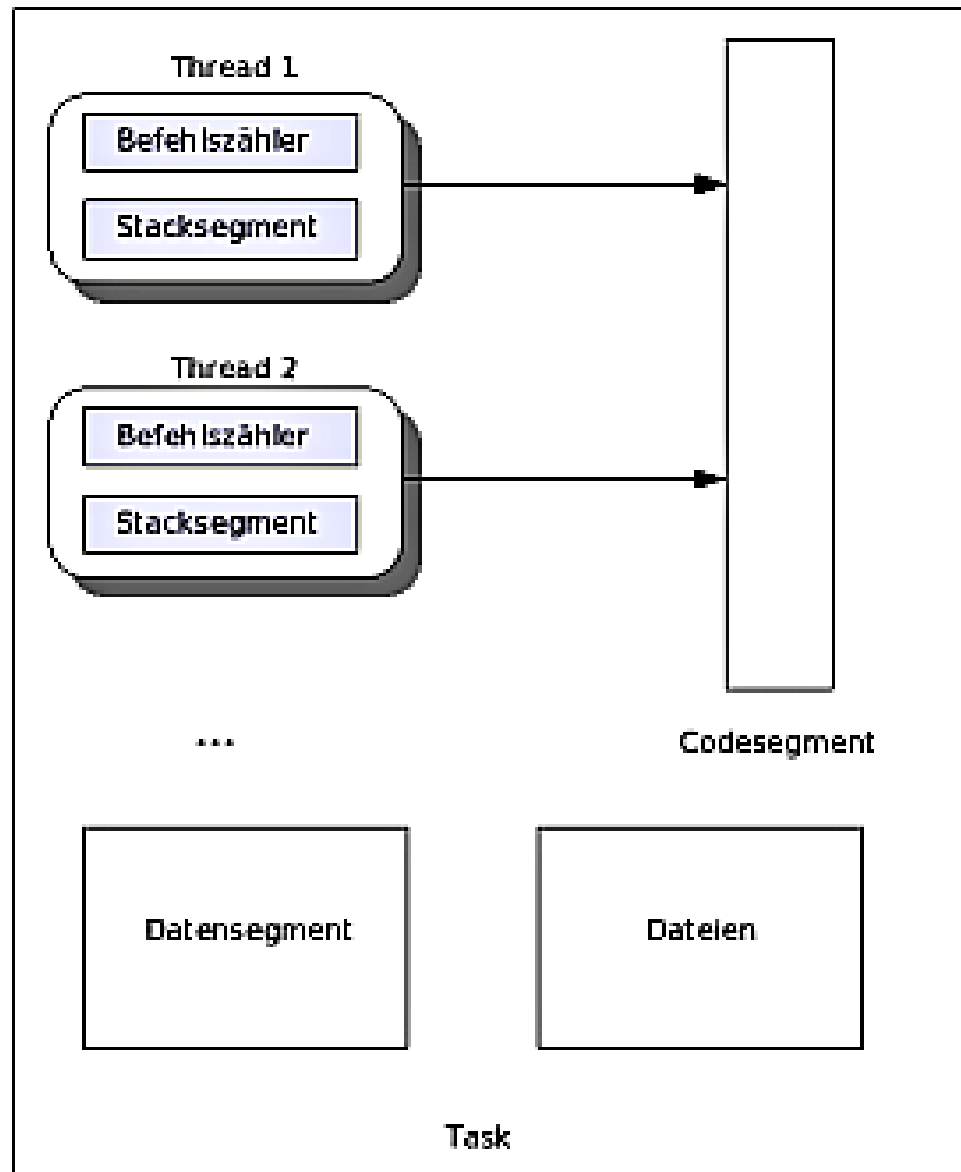
- Waise
  - Parent Prozess wurde beendet
  - Neuer Parent Prozess = Init
- Zombie
  - Beendeter Prozess, der noch als Prozess aufgelistet wird

# Threads unter Linux

- „Aktivitätsträger in der Informatik“
- Sequentieller Ablauf eines Prozesses
- Einteilung in überschaubare Einheiten

# Threads unter Linux

- Threads teilen sich Ressourcen des Prozesses (wie Adressraum, Speicher, Dateien, ...)
- Kommunikation unter Threads möglich
- Problem bei der Synchronisation (Speichernutzung)
- Geringer Aufwand – Kein Prozesskontextwechsel



<http://de.wikipedia.org/wiki/Bild:Threads.png>

# Threads unter Linux

- POSIX-Threads
  - Threads früher nicht portabel
  - POSIX-Threads neuer Standard
  - Dienen zur Multicorebenutzung
  - Mehrere Threads laufen gleichzeitig
  - In Linux früher und besser realisiert als in Windows

# Prozessverwaltung

- Drei verschiedene Schedulingmethoden
  - First in First out (FiFo)
  - Round Robin (RR)
  - Klassischer Unix Algorithmus
  - Linux benutzt den Completely Fair Scheduler  
Verwendet Einzelteile der Basisscheduler
- Proc-Dateisystem

# Prozessverwaltung

- Prozesstabelle:
  - PID = Process Identifier (ID)
  - PPID = Parent Process Identifier
  - PGID = Parent Group Identifier
  - SID = SessionID
  - Prozessübersicht mittels „-ps“
    - Anzeige aller Prozesse: -a
    - Anzeige der Abhängigkeiten: -f
    - PPID, PGID, SID und Status: -j
    - Langformat: -l
    - Zusätzliche Speicherinfos: -m
    - weitere zu sehen unter:

# Interprozesskommunikation

- Kommunikation über Ressourcen (Dateien, Peripheriegeräte, Prozesse)
  - Prozesssynchronisation
- Verschiedene Möglichkeiten:
  - Ereignisse
  - Funktionsaufrufe
  - Shared Memory

# Interprozesskommunikation

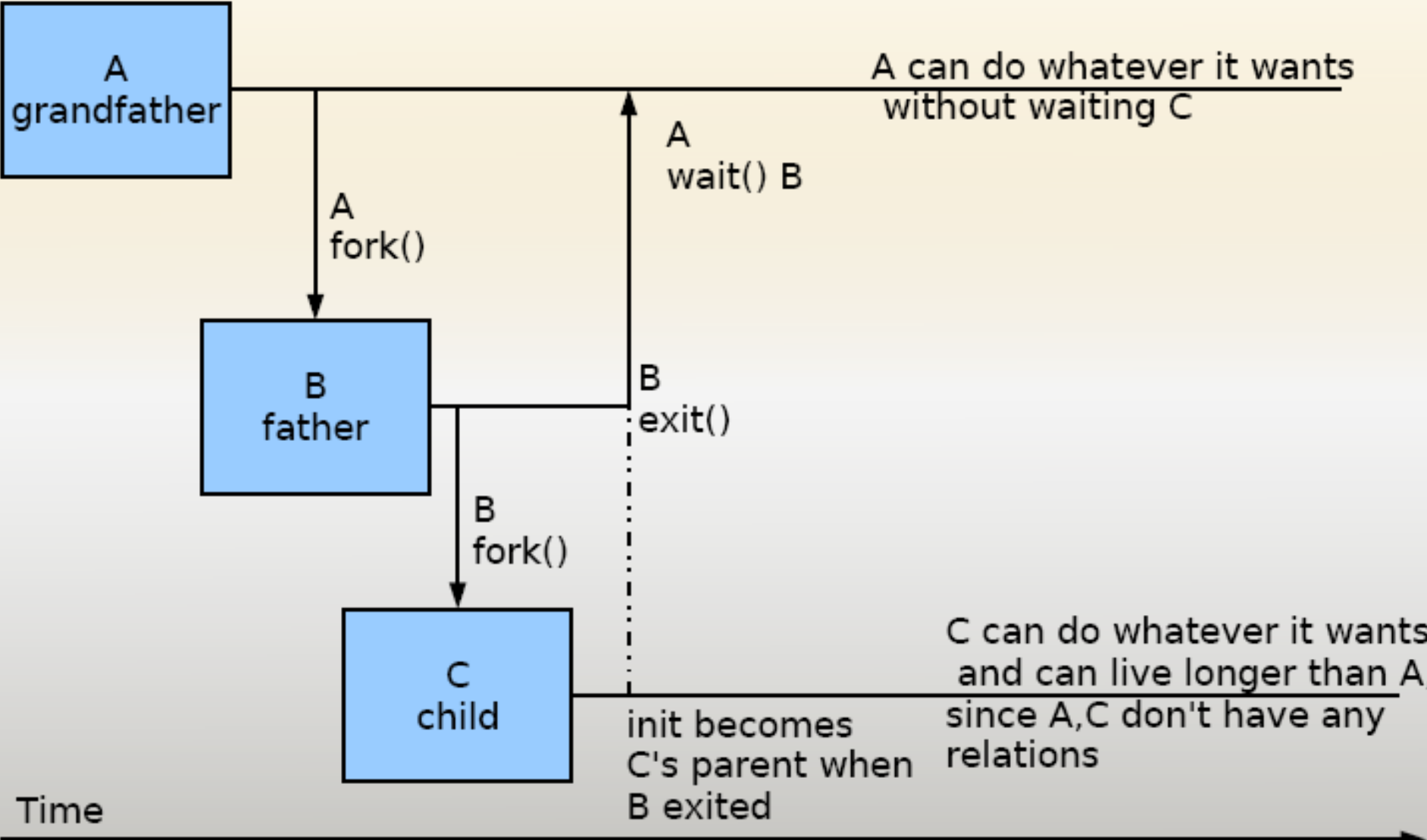
- Datenströme:
  - FiFo-Puffer
  - Pipes
  - Socket per Netzwerkprotokoll
  - File Descriptor
- Threads kommunizieren Untereinander, da sie im gleichen Adressraum arbeiten

# Der Befehl fork()

- Fork(), zu Deutsch „Gabelung“
- Kreiert einen neuen, identischen Prozess
- Alter Prozess wird als „Parent“ bezeichnet
- Neuer Prozess wird als „Child“ bezeichnet.

# Der Befehl fork()

- Daten, Maschinencode und Befehlszähler werden übernommen
- Verschiedene Speicher
- Child ist vollwertiger Prozess (eigene PID)



<http://starryalley.twbbs.org/blog/uploads/mypic/forktwice.png>

# Exec()

- Meistens in Verbindung mit fork()
- Vier verschiedene Parameter zur Ausführung:
  - e Environmentliste als Vektor erwartet
  - l Kommandorzeilenargumente als Liste
  - v Kommandozeilenargumente als Vektor
  - p Dateiname und kein Pfadname als Argument zum Aufruf eines Programms erwartet

# Exec()

- Somit sechs verschiedene exec-Befehle:
  - execl, execve, execv, execl, execvp
- Execve = Systemaufruf
- Überlagerung des Child mit Exec
- Daemons
  - Hintergrundprogramme unter Unix

# Wichtige Prozesse in Linux

- Scheduler (PID 0)
  - Regelt die Abhandlung der Prozesse/Threads
- Init (PID 1)
  - Erster Prozess der beim Booten startet
  - Erzeugt alle anderen Prozesse
  - Für Benutzerprozesse
  - Verwaltet den Runlevel
  - Nicht mehr Zeitgemäß (z.B. Ubuntu – Upstart)

# Zusammenfassung

- Prozesse bestehen aus Threads
- Threads sind einzelne Fäden die sequentiell bearbeitet werden.
- Threads benötigen keinen Kontextwechsel
- Proc-Dateisystem liefert aktuelle Daten über OS
- Prozesserzeugung mithilfe von `fork()` und `exec()`

# Quellen

- <http://de.wikipedia.org/wiki/Interprozesskommunikation>
- [http://de.wikipedia.org/wiki/Zombie \(EDV\)](http://de.wikipedia.org/wiki/Zombie_(EDV))
- [http://de.wikipedia.org/wiki/Prozess \(Informatik\)](http://de.wikipedia.org/wiki/Prozess_(Informatik))
- [http://de.wikipedia.org/wiki/Thread \(Informatik\)](http://de.wikipedia.org/wiki/Thread_(Informatik))
- <http://de.wikipedia.org/wiki/Daemon>
- [http://de.wikipedia.org/wiki/User Thread](http://de.wikipedia.org/wiki/User_Thread)
- [http://de.wikipedia.org/wiki/Fork \(Unix\)](http://de.wikipedia.org/wiki/Fork_(Unix))
- <http://de.wikipedia.org/wiki/Runlevel>
- <http://de.wikipedia.org/wiki/Prozesskontext>
- [http://medien.informatik.uni-ulm.de/lehre/courses/ss02/linux/22\\_proseminar\\_linux\\_100702.pdf](http://medien.informatik.uni-ulm.de/lehre/courses/ss02/linux/22_proseminar_linux_100702.pdf)
- <http://www.pronix.de/pronix-156.html>
- <http://de.wikipedia.org/wiki/Upstart>