

LUG Ahaus



<http://lugah.de>

# Linux- Der Kernel 2.6

## Der Linux Kernel Ver. 2.6

Linux User Group Ahaus

Jan. 04

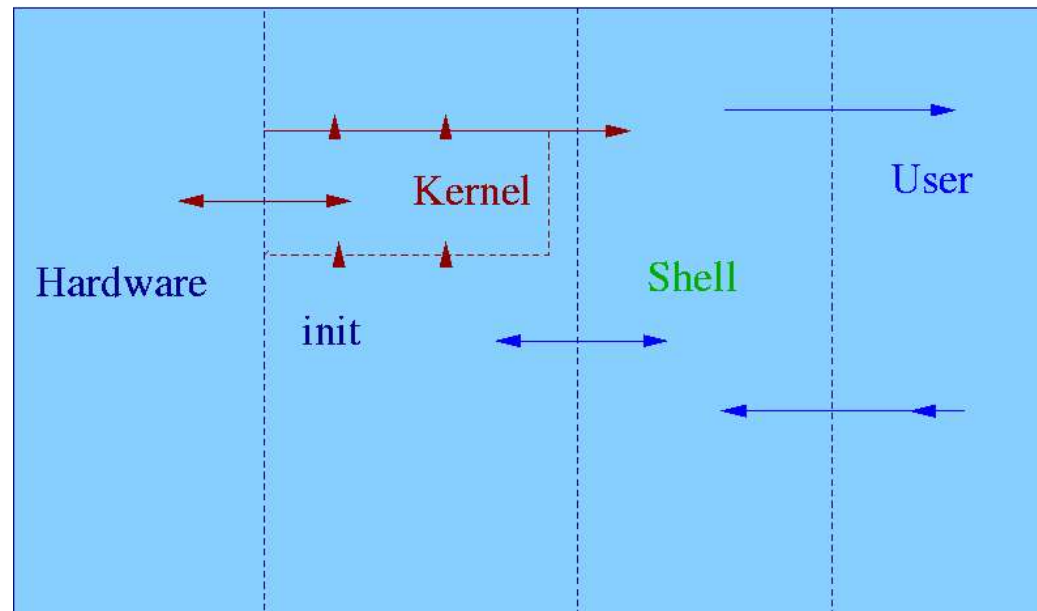
Rainer Ostendorf  
[rainer@lugah.de](mailto:rainer@lugah.de)



# Linux- Der Kernel 2.6

## Kernel: Was ist das jetzt eigentlich?

Grob: Die Software, die die Schnittstelle zwischen Hardware und Anwenderprogrammen bildet.





# Linux- Der Kernel 2.6

## Kernel: Was ist das jetzt eigentlich?

Konkret: Der Kernel ist /bzImage im Dateisystem

Aufgaben:

- Hardwareabstraktion
- Zugriffsregelung für gemeinsame Ressourcen wie:  
Prozessor, Speicher, Dateisystem, Peripherie
- Prozessverwaltung und Scheduling
- Speicherverwaltung



# Linux- Der Kernel 2.6

## Kernel: Grundlegendes

- Der Kernel enthält neben fundamentalen Dingen wie Scheduler, virtueller Speicherverwaltung oder Dateisystemverwaltung z.B. auch den IP-Stack oder Treiber für die installierte Hardware.
- Man kann einzelne Komponenten des Kernel als sog. Modul konfigurieren. Modules lassen sich zur Laufzeit laden.  
->Vorteil: Schlankeres System, flexibler.



# Linux- Der Kernel 2.6

## Das neue SYS-FS

- Einheitliches Gerätemodell
- Einteilung Busse, Geräte, Klassen (Eingabe, Netzwerk, TTYs)
- Alle Geräte in Liste von kObjectStrukturen
- Zugang Pseudodateisystem:  

```
sysfs mount -t sysfs none /sys
```
- Name, IRQ, DMA, Status Stromversorgung
- /proc nur noch für Prozessinformationen (so der Plan)
- Verlagerung von Hardware Wissen von Treiber in Kernel
  - > Vereinfachung der Systemkonfiguration
  - > Notwendig für moderne Features: ACPI, Hotplug



# Linux- Der Kernel 2.6

## O(1) Scheduler

- Bisher: Prozessliste, wird immer länger, je mehr Prozesse
- Liste muss durchsucht werden um den nächsten Prozess der an der Reihe ist zu finden -> Taskswitch wird mit zunehmender Prozesszahl langsamer
- Das ist Mist. Besser: der neue O(1) Scheduler:
  - Prozesse stecken in einem Array aus 140 verketteten Listen
  - Davon 0-99 für Prozesse mit Echtzeit-Priorität, Rest für nice-Werte von -20 bis 20
  - In einer Bitmap zeigen 140 Bits an, welche Liste lafbereite Prozesse enthält
  - Nächster Task: Lediglich erstes gesetztes Bit finden
  - Von den Arrays gibt es 2 Stück, nach Verbrauch der Rechenzeit wird ein Prozess von der einen in die andere bewegt.
  - Wenn das Array mit aktive Tasks leer ist wird getauscht.
  - Bei SMP: Jede CPU hat so eine Runqueue, bei >25% Unterschied wird ausgeglichen



# Linux- Der Kernel 2.6

## Präemptives Multitasking

- Bis 2.4: Scheduler liegt auf Eis, solange Kernel-Funktionen ausgeführt werden
- Jetzt ist auch die Mehrzahl der Kernel-Funktionen unterbrechbar, nur noch wenige Funktionen sind mit Spinlocks geschützt
- Aufteilung längerer kritischer Funktionen in durch „preemption points“ getrennte Bereiche
- Verbesserung der Latenz bei Interrupts um bis zu Faktor 10(!) zwischen 2.4 und 2.6
- Wichtig im Embedded Bereich, aber auch bei Desktop-Anwendungen, z.B. dem Dekodieren von MPEG-Strömen in Echtzeit
-



# Linux- Der Kernel 2.6

## Speicherverwaltung

- Im VM ist der rmap-Patch von Rik van Riel integriert: bisher ist es nur erlaubt gewesen die logisch Adresse zu einer physischen zuzuordnen.
- 2.6: PTE(PageTableEntry)-Liste: Zuordnung von Prozessen zu jeder Speicherseite
- Dadurch Beschleunigung bei Freigabe bei Speicher: Man muss nicht den Speicher eines beendeten Prozesses zusammensuchen, sondern kann gezielt nachschauen, welche Seiten er belegt hatte.
- Besonders von Vorteil bei Maschinen mit sehr viel Speicher und bei Speicherknappheit
- SGI hat viel NUMA(NonUniformMemoryAccess)-Code beigesteuert:
- Toll wenn man seinen Speicher verteilen will, noch toller wenn man solche Hardware überhaupt mal hat... :)



# Linux- Der Kernel 2.6

## Threading

- NPTL(NativePosixThreadingLibrary) löst die alten LinuxThreads ab
- Die alte Thread-Implementierung hat Skalierungsprobleme -> schlecht für grosse Firmenanwendungen mit vielen Threads
- Früher: Maximal 8192 Threads, jetzt theoretisch keine Beschränkung mehr, über 100.000 Threads sind scheinbar demonstriert worden
- Früher: Erzeugen eines Threads dauert um so länger, je mehr Threads es gibt, jetzt: `sys_clone()`-Aufruf erzeugt Thread in konstanter Zeit
- FUTEXE (Fast Userspace Mutexe) machen die Synchronisation von Threads effizienter: Man kann sich solange schlafen legen bis ein Speicherbereich geändert wurde, dann automatisches „Wecken“ -> das erspart regelmäßiges Abfragen oder Kommunikation über Signale
- Futexe mittlerweile allgemeines Mittel der Synchronisation von Userspace-Prozessen mit Kernel-Unterstützung



# Linux- Der Kernel 2.6

## Kernel 2.6: HyperThreading

- Alias Simultaneous Multithreading (SMT)
- Maskierung eines Prozessors als 2 oder mehr Prozessoren
- Dadurch wird der Scheduler komplexer: Er muss eine Optimierung der CPU-Last auf virtuelle und reale Prozessoren vornehmen



# Linux- Der Kernel 2.6

## I/O-System

- Subsystem für blockorientiertes I/O (also z.B. Festplatten) wurde von Grund auf neugeschrieben. Das galt schon lange als nötig
- Das war der Grund warum 2.5 lange Zeit unbenutzbar war (es waren über eine halbe Mio. Code-Zeilen)
- Insgesamt sind Lese- und Schreiboperationen schneller geworden
- Grenzen sind gefallen: 32bit: 16TByte, 64bit 8ExaByte(!) - man hat definitiv keine Probleme mit der Grösse der heimischen MP3-Sammlung
- LVM2 (Logical Volume Manager) ist im Kernel integriert, damit lassen sich z.B. mehrere Festplatten zu einem logischen Laufwerk zusammenfassen. Kein Platz mehr: Platte dazuschieben
- Atapi-Brenner werden direkt unterstützt, keine schwindelige SCSI-Emulation mehr.
- Die Treiberprogrammierung wird einfach, damit wenige fehleranfällig



# Linux- Der Kernel 2.6

## Viele bunte Architekturen

- Es sind Grosse Teile des uLinux-Projektes in den Kernel eingeflossen
- Linux rennt jetzt auf den kleinen schwarzen Käfern mit den vielen silbernen Beinen dran (aber ohne eigene MMU)
- Das Zeitverhalten ist vorhersagbarer geworden (weiche Echtzeit anwendungen)
- Auslagern von Speicherseiten auf die Platte lässt sich komplett deaktivieren -> keine page faults, denn die dauern ewig
- Bald hat meine Waschmaschine Maus und Tastatur...



# Linux- Der Kernel 2.6

## Modulsystem

- Modulsystem ist komplett verändert.
- Es gibt neue mod-utils. Ist auch besser so, denn die alten funktionieren jetzt nicht mehr. Die neuen Tools heissen Module-Init-Tools.
- Äusserlich sind neue Module an der Endung .ko erkennbar (für KernelObject)
- Neue Konfigdatei namens /etc/modprobe.conf löst die /etc/modules.conf ab
- Skript generate\_modprobe.conf konvertiert aber netterweise.
- Trotzdem etwas Handarbeit: Einige Module haben neue Namen, z.B. uhci heisst jetzt uhci-hcd (für Host Control Driver)



# Linux- Der Kernel 2.6

## Konfiguration

- Neues Konfigurationssystem. Die alten Config.in-Dateien sind weg, dafür gibt's jetzt Kconfig-Dateien, die für die Quelltexte im Verzeichnis jeweils die möglichen Optionen, Typen und Erläuterungen enthalten.
- Die zentrale Hilfedatei /Documentation/Configure.help ist verschwunden
- Es empfiehlt sich in Documentation/changes zu schauen, da steht drin, welche Software in welche Version nötig ist.
- Grundkonfiguration aus 2.4er: **make oldconfig**. Anschliessende Durchsicht, ob auch alles passt.
- **make xconfig** bringt einen graphischen Dialog in QT, **make gconfig** einen in Gtk+
- Aber **make menuconfig** geht auch weiterhin (zum Glück :)



# Linux- Der Kernel 2.6

## Kompilierung

- **make dep** ist nicht mehr notwendig
- Ein einfaches **make** erzeugt ein komprimiertes Image (bzImage), das unkomprimierte Image (vmlinuz) sowie die module
- Module lassen sich wie gehabt mit `make modules_install` installieren
- Es fliegen einem nicht mehr sämtliche Details des Build-Prozesses um die Ohren, die Ausgabe beschränkt sich auf das wesentliche. Wer leiden will nimmt **make V=1**
- **make help** zeigt alle Build-Ziele an



# Linux- Der Kernel 2.6

## Installation

- Die Standard-Prozedur (für i386er):
  - `make menuconfig` #konfigurieren
  - `make` #kompilieren
  - `make modules_install` #Module installieren
  - `cp /bzImage /bzimage_old` #Kernel-Image sichern
  - `cp /usr/src/linux/arch/i386/boot/bzImage /` #Image kopieren
  - `cp /boot/System.map /boot/System.map.old` #SystemMap sichern
  - `cp System.map /boot` #System-Map kopieren
  - `/etc/lilo.conf` anpassen
  - `lilo` rufen(!)
  - rebooten
  - Spass haben

LUG Ahaus



<http://lugah.de>

# Linux- Der Kernel 2.6

Das wars!

Vielen Dank für eure Aufmerksamkeit.

Quellen: Dr. Oliver Diedrick, c't 24/03, Artikel S.194,  
<http://www.selflinux.org>