

Linux Kernelverwaltung

Christopher Pommrenke

Linux für Umsteiger

06.05.2008

Gliederung

- Aufbau des Linux Kernels
- Update eines Kernels
- Verwaltung mehrerer Kernel
- Kernel Module und ihre Verwaltung

Aufbau: Versionsnummern

- Versionsnummer besteht aus 4 Ziffern
 - 1. Ziffer: Version der grundlegenden Architektur
 - 2. Ziffer: Majorrelease
 - 3. Ziffer: Minorrelease
 - 4. Ziffer: Bei Korrekturen kritischer Fehler
- Versionsabfrage mit *uname -a*

Aufbau: Dateien des Kernels

- Kernel-Image:
 - Liegt in: */boot*
- Initial RAM disk:
 - Liegt in: */boot*
- Kernel-Header:
 - Liegen in: */usr/src*
- Module:
 - Liegen in: */lib/modules/kernel-version*

Aufbau: Module

- Nicht direkt in Kernel eingebunden (linked)
- Erhöhen Flexibilität
- Halten den Kernel klein und übersichtlich
- Vereinfachen Konfiguration
- Sparen Speicher
- Beispiele: z.B. Netzwerktreiber, Dateisysteme

Gliederung

- Aufbau des Linux Kernels
- **Update eines Kernels**
- Verwaltung mehrerer Kernel
- Kernel Module und ihre Verwaltung

Kernel Update: Gründe

Pro:

- Unterstützung für Hardware eingefügt
- Sicherheit verbessert
- Geschwindigkeit erhöht
- Experimentierfreudigkeit
- Spezielle Software benötigt eigenen Kernel

Contra:

- Hoher Aufwand
- Bringt nicht immer Verbesserungen
- Kompatibilitätsprobleme
- Belegt unnötig Platz

Kernel Update: Vorbereitung

- Backup wichtiger Dateien
- Manuelle Konfigurationen durchgeführt?
- Neuen kompilierten Kernel besorgen

Kernel Update: Vergleich

OpenSuse

- Installation mittels rpm
- Aktuell installierte Version:
rpm -qa 'kernel'*
- Installieren:
rpm -ihv neuer-kernel.rpm
- Bei Problemen:
rpm -qpl neuer-kernel.rpm
- Eventuell: *--force*

Ubuntu

- Installation mittels apt
- Paketname:
linux-image-VERSION-SUBVERSION-ARCHITEKTUR
- Besser Metapaket:
linux-ARCHITEKTUR
- Beispiel:
apt-get install linux-generic

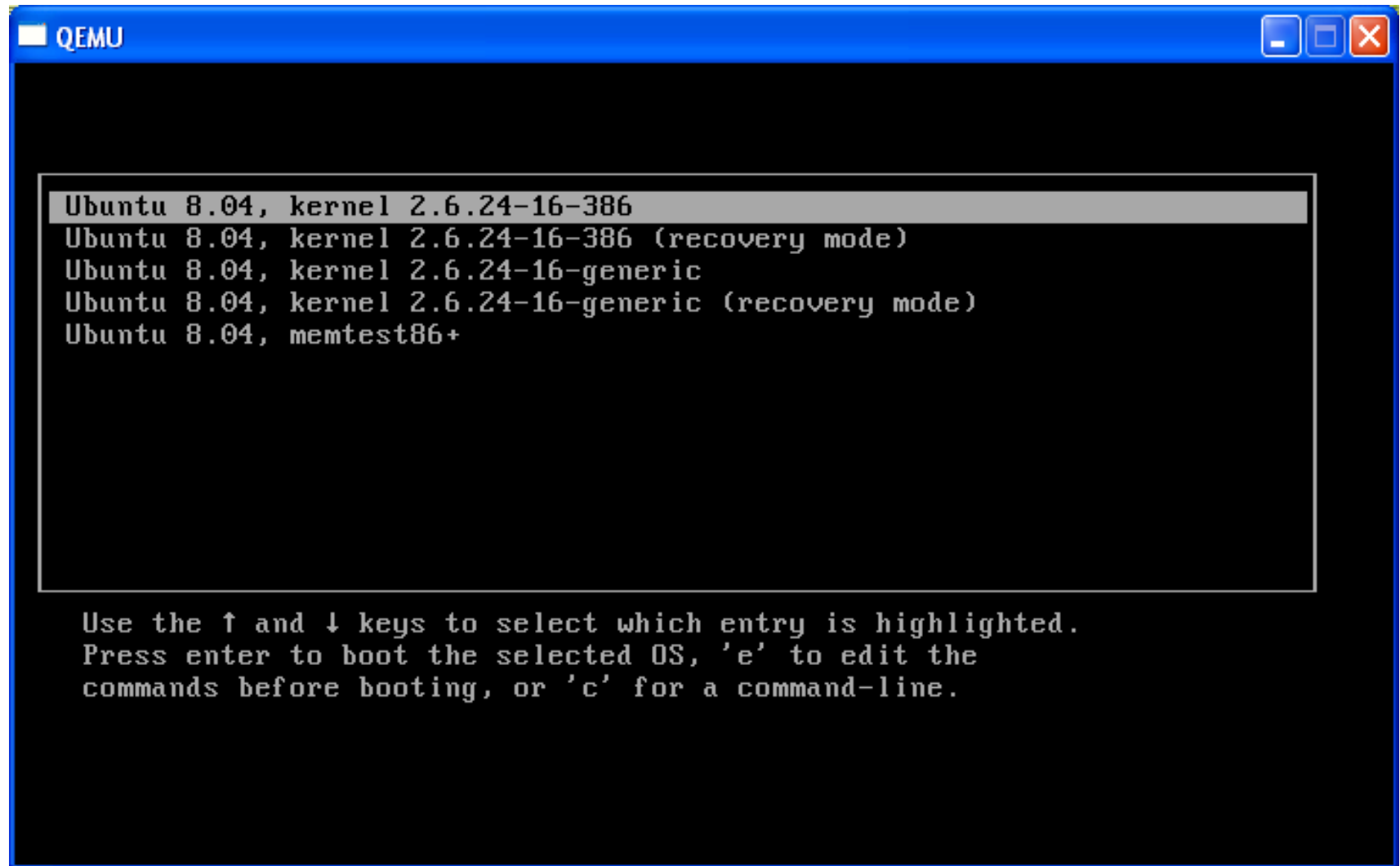
Kernel Update: Und dann?

- Neuen Kernel testen
- Nicht mehr benötigte Kernel entfernen
- Manuell installierte Module erneut installieren
- Bootloadereinträge überprüfen

Gliederung

- Aufbau des Linux Kernels
- Update eines Kernels
- **Verwaltung mehrerer Kernel**
- Kernel Module und ihre Verwaltung

mehrere Kernel: Bootmenü



```
QEMU

Ubuntu 8.04, kernel 2.6.24-16-386
Ubuntu 8.04, kernel 2.6.24-16-386 (recovery mode)
Ubuntu 8.04, kernel 2.6.24-16-generic
Ubuntu 8.04, kernel 2.6.24-16-generic (recovery mode)
Ubuntu 8.04, memtest86+

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.
```

mehrere Kernel: Grub

- Konfigurationsdatei: *boot/grub/menu.lst*
- Enthält:
 - Konfiguration für Grub
 - Informationen für update-grub
 - Kommentare
- Bootparameter können beim Starten dem Kernel übergeben werden

Gliederung

- Aufbau des Linux Kernels
- Update eines Kernels
- Verwaltung mehrerer Kernel
- **Kernel Module und ihre Verwaltung**

Module: Verwaltung

- Anzeigen der geladenen Module: *lsmod*
- Modulinformationen: *modinfo*
- Modul Laden: *insmod MODULNAME*
- Modul entfernen: *rmmmod MODULNAME*
- Besser: *modprobe MODULNAME*
- Wichtig beim Laden: Modulparameter

Module: Abhängigkeiten

- Module setzen eventuell andere voraus
- Alle Module & Abhängigkeiten gespeichert in:
/lib/modules/kernel-version/modules.dep
- modules.dep aktualisieren: *depmod*

Module: Wissenswertes

- Module beim booten automatisch laden:
/etc/modules
- Automatisches laden eines Moduls verhindern:
/etc/modprobe.d

Zusammenfassung

- Kernel wird ständig weiter entwickelt
- Installation eines Kernels mittlerweile einfach
- Module sind wichtiger Zusatz zum Kernel

Quellen

- <http://www.galileocomputing.de/openbook/ubuntu/index.htm>
- <http://www.thomashertweck.de/kernel26.html#rpm>
- <http://www.ibm.com/developerworks/linux/library/l-initrd.html>
- <http://wiki.ubuntuusers.de/menu.lst>
- <http://wiki.ubuntuusers.de/Kernel#head-a399dc37c97ec061381b5480de713edf740d0762>
- <http://strcat.neessen.net/eigenes/module.html>
- http://de.wikipedia.org/wiki/Linux_Kernel

Linux Kernelverwaltung

Christopher Pommrenke

Linux für Umsteiger

06.05.2008

Gliederung

- **Aufbau des Linux Kernels**
- Update eines Kernels
- Verwaltung mehrerer Kernel
- Kernel Module und ihre Verwaltung

Aufbau: Versionsnummern

- Versionsnummer besteht aus 4 Ziffern
 - 1. Ziffer: Version der grundlegenden Architektur
 - 2. Ziffer: Majorrelease
 - 3. Ziffer: Minorrelease
 - 4. Ziffer: Bei Korrekturen kritischer Fehler
- Versionsabfrage mit *uname -a*

Beispiel: Aktuelle Version

Ziffer 1: wird nur bei grundlegenden Änderungen der Systemarchitektur angehoben (aktuell: 2)

Ziffer 2: hier stehen gerade Zahlen für stabile Versionen, während die ungeraden Zahlen für Testversionen zur Entwicklung stehen. Allerdings ist dieses System seit 2004 ausgesetzt, Änderungen werden seither direkt in die aktuelle 2.6er Serie eingearbeitet

Ziffer 3: kennzeichnet die eigentliche Version, sie wird erhöht wenn neue Funktionen in den Kernel eingebunden werden

Ziffer 4: im März 2005 wurde diese Ziffer erstmals eingeführt. Sie wird erhöht wenn Fehler im Kernel beseitigt werden, ohne das neue Features hinzukommen

`uname -a` liefert folgende Informationen:

Kernelname, Hostname, Kernelversion, Kompilierdatum, Computerarchitektur, Betriebssystem

Aufbau: Dateien des Kernels

- Kernel-Image:
 - Liegt in: */boot*
- Initial RAM disk:
 - Liegt in: */boot*
- Kernel-Header:
 - Liegen in: */usr/src*
- Module:
 - Liegen in: */lib/modules/kernel-version*

Kernel-Image: ist die Datei die den fertig kompilierten Kernel enthält. Standardnamen sind: *vmlinux* (unkomprimiert) *vmlinuz* (komprimiert). Außerdem gibt es noch *zImage* (früher) und *bzImage* wobei *b* für *big* steht und eingeführt wurde als der Kernel immer größer wurde. Letztendlich ist der Name eines Kernels aber nur ein Name und kann daher frei Umbenannt werden

Initial RAM disk: ist ein temporäres *root*-Verzeichnis, das beim booten im Arbeitsspeicher angelegt wird bevor das eigentliche *root*-Verzeichnis erreichbar ist. Es enthält diverse Treiber und Programme die es erlauben das echte *root*-Verzeichnis zu mounten und z.B. Module zu laden. Somit müssen nicht alle Funktionen fest in den Kernel einkompiliert werden, was ihn erheblich kleiner macht.

Kernel-Header: werden benötigt, um Programme aus ihren *sources* zu kompilieren

Module: im Moduleverzeichnis befinden sich weitere Unterordner für die verschiedenen Typen von Modulen z.B. für Datei Systeme oder für Grafikkarten

Aufbau: Module

- Nicht direkt in Kernel eingebunden (linked)
- Erhöhen Flexibilität
- Halten den Kernel klein und übersichtlich
- Vereinfachen Konfiguration
- Sparen Speicher
- Beispiele: z.B. Netzwerktreiber, Dateisysteme

Ohne Module muss für jede neue Funktion (z.B. verwendung einer neuen Netzwerkkarte) der Kernel neu kompiliert werden. Module können im laufenden Betrieb geladen und ausgetauscht werden sie werden im Normalfall dynamisch dann geladen wenn sie benötigt werden

Gliederung

- Aufbau des Linux Kernels
- **Update eines Kernels**
- Verwaltung mehrerer Kernel
- Kernel Module und ihre Verwaltung

Kernel Update: Gründe

Pro:

- Unterstützung für Hardware eingefügt
- Sicherheit verbessert
- Geschwindigkeit erhöht
- Experimentierfreudigkeit
- Spezielle Software benötigt eigenen Kernel

Contra:

- Hoher Aufwand
- Bringt nicht immer Verbesserungen
- Kompatibilitätsprobleme
- Belegt unnötig Platz

Man sollte sich vor einem Kernel-Update genau überlegen welche Verbesserungen man sich erhofft und was gegen ein Update spricht um dies gegeneinander abzuwägen
Lustre z.B. ist ein Cluster-Dateisystem das seinen eigenen Kernel benötigt

Kernel Update: Vorbereitung

- Backup wichtiger Dateien
- Manuelle Konfigurationen durchgeführt?
- Neuen kompilierten Kernel besorgen

Auch wenn heutzutage ein Kernel-Update eigentlich ohne Probleme verläuft, sollte man sich trotzdem bewusst machen, dass man den neuen Kernel als Root installiert und somit bei auftretenden Problemen sehr schnell großen Schaden anrichten kann.

Quelle für OpenSuse-Kernel:

<ftp://ftp.suse.com/pub/projects/kernel/kotd/>

Von Hand installierte zusätzliche Module z.B. müssen für jeden Kernel neu installiert werden.

Beispiele für manuell installierte Pakete sind z.B.

Grafikkartentreiber von ATI oder Nvidia, da diese keine Open Source Treiber sind und somit meistens nicht automatisch installiert wurden

Ebenso kann es sein das z.B. die Konfigurationsdatei des Bootmanagers geändert wurde

Kernel Update: Vergleich

OpenSuse

- Installation mittels rpm
- Aktuell installierte Version:
rpm -qa 'kernel'*
- Installieren:
rpm -ihv neuer-kernel.rpm
- Bei Problemen:
rpm -qpl neuer-kernel.rpm
- Eventuell: *--force*

Ubuntu

- Installation mittels apt
- Paketname:
linux-image-VERSION-SUBVERSION-ARCHITEKTUR
- Besser Metapaket:
linux-ARCHITEKTUR
- Beispiel:
apt-get install linux-generic

Ubuntu: bis Ubuntu 6.06 gab es verschiedene Kernelarchitekturen für die einzelnen Systeme (z.B. -i386), seit Ubuntu 6.10 gibt es nur noch einen Kernel (-generic) der für die meisten Architekturen funktioniert

Metapaket: bei einer größeren Installation für die mehrere Pakete benötigt werden sind die Paketinformationen in einem einzigen Metapaket zusammengefasst

rpm -qpl neuer-Kernel.rpm liefert einem Informationen welche Dateien installiert werden sollen, somit kann man auftretende Probleme überprüfen und wenn diese nicht gravieren erscheinen mit dem rpm Zusatz *--force* die auftretenden Konflikte ignorieren

Wer bei OpenSuse YaST zum installieren eines neuen Kernel verwenden möchte muss nur das Suse-KOTD-Verzeichnis als zusätzliche Installationsquelle angeben. Vorsicht, hierbei handelt es sich nicht um eine Paralellinstallation, der alte Kernel wird gelöscht!

Kernel Update: Und dann?

- Neuen Kernel testen
- Nicht mehr benötigte Kernel entfernen
- Manuell installierte Module erneut installieren
- Bootloadereinträge überprüfen

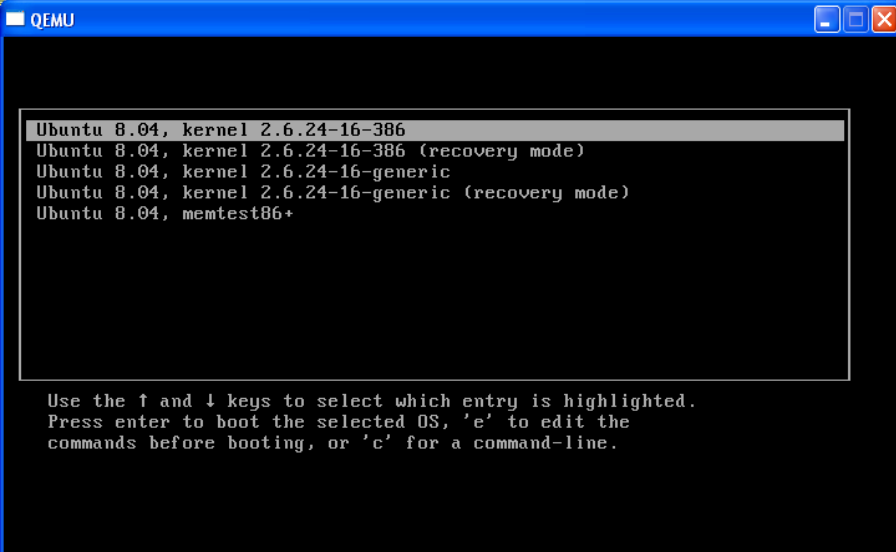
Da bei einem Kernel-Update der alte Kernel im normalfall nicht gelöscht wurde kann man bei Problemen einfach den alten Kernel booten

alte und nicht funktionierende Kernel kann man wie bei der Installation ebenfalls mit er Paketverwaltung deinstallieren.
Vorsicht bei Ubuntu: nicht das Metapaket sondern nur das Paket: *linux-image-VERSION-SUBVERSION-ARCHITEKTUR* entfernen

Gliederung

- Aufbau des Linux Kernels
- Update eines Kernels
- **Verwaltung mehrerer Kernel**
- Kernel Module und ihre Verwaltung

mehrere Kernel: Bootmenü



```
QEMU
-----
Ubuntu 8.04, kernel 2.6.24-16-386
Ubuntu 8.04, kernel 2.6.24-16-386 (recovery mode)
Ubuntu 8.04, kernel 2.6.24-16-generic
Ubuntu 8.04, kernel 2.6.24-16-generic (recovery mode)
Ubuntu 8.04, memtest86+

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.
```

Hier kann man dem Kernel z.B. Bootparameter übergeben. Dazu muss der zu startende Booteintrag ausgewählt werden, mit der Taste e kommt man in ein Untermenü in dem man den Eintrag Kernel markieren muss um dann erneut die Taste e zu drücken, damit man nun am ende der Zeile den gewünschten Bootparameter eintragen kann. Ein Druck auf Enter und anschließend der b Taste bootet nun das System mit den gewünschten Optionen

mehrere Kernel: Grub

- Konfigurationsdatei: *boot/grub/menu.lst*
- Enthält:
 - Konfiguration für Grub
 - Informationen für update-grub
 - Kommentare
- Bootparameter können beim Starten dem Kernel übergeben werden

menu.lst: Zeilen mit 2 oder mehr Rauten sind Kommentare, Zeilen mit einer # sind Anweisungen für update-grub, falls bestimmte Schlüsselwörter folgen, ansonsten ebenfalls kommentare. Zeilen ohne Raute sind optionn für grub.

Wichtige Einträge:

- default: Nummer des Eintrags der automatisch geladen wird wenn keine Eingabe erfolgt
- timeout: zeit bis default Eintrag geladen wird
- hiddenmenu: Bootmenü anzeigen oder nur mit ESC zu erreichen

Beispiele für Bootparameter sind:

noacpi deaktiviert die automatische acpi-Erkennung

ht=on aktiviert hyper-threading

BOOT_DEBUG=2|3 Fehlersuche während des bootvorgangs

Gliederung

- Aufbau des Linux Kernels
- Update eines Kernels
- Verwaltung mehrerer Kernel
- **Kernel Module und ihre Verwaltung**

Module: Verwaltung

- Anzeigen der geladenen Module: *lsmod*
- Modulinformationen: *modinfo*
- Modul Laden: *insmod MODULNAME*
- Modul entfernen: *rmmmod MODULNAME*
- Besser: *modprobe MODULNAME*
- Wichtig beim Laden: Modulparameter

Mit dem Befehl *modprobe* werden beim Laden eines Moduls auch eventuelle Abhängigkeiten aufgelöst und zusätzlich benötigte Module automatisch mitgeladen. Ebenso werden beim entfernen eines Moduls alle Module mitentfernt die nicht mehr benötigt werden

Mögliche Modulparameter kann man mit *modinfo* in erfahrung bringen

Module: Abhängigkeiten

- Module setzen eventuell andere voraus
- Alle Module & Abhängigkeiten gespeichert in: */lib/modules/kernel-version/modules.dep*
- *modules.dep* aktualisieren: *depmod*

Aus der Datei *modules.dep* entnimmt *modprobe* seine Informationen zum auflösen der Abhängigkeiten beim Laden *modules.dep* sollte immer auf dem aktuellen Stand sein, *depmod* wird deshalb normalerweise beim Systemstart ausgeführt

Module: Wissenswertes

- Module beim booten automatisch laden:
/etc/modules
- Automatisches laden eines Moduls verhindern:
/etc/modprobe.d

/etc/modules ist eine Datei in die man Modulnamen eintragen kann, damit diese beim Systemstart automatisch geladen werden

/etc/modprobe.d ist ein Ordner in dem Blacklistdateien gespeichert sind. Möchte man verhindern, dass ein Modul automatisch geladen wird, muss man eine Datei mit dem Namen *blacklist-XYZ* erstellen und in diese den Eintrag *blacklist modulname* einfügen

Zusammenfassung

- Kernel wird ständig weiter entwickelt
- Installation eines Kernels mittlerweile einfach
- Module sind wichtiger Zusatz zum Kernel

Quellen

- <http://www.galileocomputing.de/openbook/ubuntu/index.htm>
- <http://www.thomashertweck.de/kernel26.html#rpm>
- <http://www.ibm.com/developerworks/linux/library/l-initrd.html>
- <http://wiki.ubuntuusers.de/menu.lst>
- <http://wiki.ubuntuusers.de/Kernel#head-a399dc37c97ec061381b5480de713edf740d0762>
- <http://strcat.neessen.net/eigenes/module.html>
- http://de.wikipedia.org/wiki/Linux_Kernel