Kernel-based Virtual Machine (KVM)

René Pfeiffer <pfeiffer@luchs.at>

CaT

17. April 2009

Inhaltsübersicht - Wovon reden wir?

Inhaltsübersicht - Wovon reden wir?

- Warum KVM?
- Wie lange noch? - Softwarepatente
- Virtualiserung mit KVM
- Aufbau / Eigenschaften / *Look & Feel*
- Administration
- Beispiele für Verwendung von KVM

Unvollständige Geschichte der Virtualisierung

Unvollständige Geschichte der Virtualisierung

- Dissertation von Fritz Rudolf Güntsch (1957)
- experimentell auf IBM M44/44X (nach 1961)
- präsent im IBM System 360 (1964 eingeführt)
- verbessert im IBM System 370 (1970 eingeführt)
- Distributed Computing ersetzt Virtualisierung (teilweise)
- Versuch der Virtualisierung auf Intel x86 (ab 1990)
- VMware® Workstation erscheint 1999
- Xen™, Parallels, VirtualBox, KVM, ...

Werbeeinblendung - VMware®

> *. . . Greene flat out dismissed our proposal that the company fly the freak flag by open sourcing its flagship code. „There is still a lot of innovation going into our hypervisor," Greene told us. „As long as there is a lot of innovation going in, (open source) is not the right model." . . .*

VMware® CEO Diane Greene im Interview mit The Register® UK am 2. November 2007

Mit *innovation* meint Diane in Wirklichkeit *Softwarepatente*, die möglicherweise eingesetzt werden, wenn VMware® mehr Konkurrenz bekommt. Das Patentieren von Technologien, die es seit über 40 Jahren gibt, ist sehr mutig. Applaus!

.

Mögliche Virtualisierungstechniken

- Citrix Xen™
- Microsoft® Hyper-V
- Parallels Workstation
- Linux-VServer
- OpenVZ
- QEMU
- Red Hat Kernel-based Virtual Machine (KVM)
- Sun Virtualbox
- VMware® Produkte

Es gibt noch weitere Projekte und Produkte.

.

Warum KVM?

Warum KVM?

- 100% Freie Software (trotz enthaltener Innovation)

Warum KVM?

- 100% Freie Software (trotz enthaltener Innovation)
- absolut schmerzfreies Installieren und Verwalten
  - `./configure && make && sudo make install`
  - geht mit fast jeder aktuelleren Distribution
  - Host kann x86_64- oder IA32-basiert sein
  - „headless management” möglich (SSH Login statt KDE/GNOME Bloat)
  - Gäste kann man skriptbasiert steuern

Warum KVM?

- 100% Freie Software (trotz enthaltener Innovation)
- absolut schmerzfreies Installieren und Verwalten
  - `./configure && make && sudo make install`
  - geht mit fast jeder aktuelleren Distribution
  - Host kann x86_64- oder IA32-basiert sein
  - „headless management" möglich (SSH Login statt KDE/GNOME Bloat)
  - Gäste kann man skriptbasiert steuern
- unterstützt Hardware-Virtualisierung (Intel VT-x & AMD-V)

Warum KVM?

- 100% Freie Software (trotz enthaltener Innovation)
- absolut schmerzfreies Installieren und Verwalten
  - `./configure && make && sudo make install`
  - geht mit fast jeder aktuelleren Distribution
  - Host kann x86_64- oder IA32-basiert sein
  - „headless management” möglich (SSH Login statt KDE/GNOME Bloat)
  - Gäste kann man skriptbasiert steuern
- unterstützt Hardware-Virtualisierung (Intel VT-x & AMD-V)
- Linux® ist Hypervisor

Warum KVM?

- 100% Freie Software (trotz enthaltener Innovation)
- absolut schmerzfreies Installieren und Verwalten
  - `./configure && make && sudo make install`
  - geht mit fast jeder aktuelleren Distribution
  - Host kann x86_64- oder IA32-basiert sein
  - „headless management” möglich (SSH Login statt KDE/GNOME Bloat)
  - Gäste kann man skriptbasiert steuern
- unterstützt Hardware-Virtualisierung (Intel VT-x & AMD-V)
- Linux® ist Hypervisor
- ¡keine Probleme mit Zeitsynchronisation!

Warum KVM?

- 100% Freie Software (trotz enthaltener Innovation)
- absolut schmerzfreies Installieren und Verwalten
  - `./configure && make && sudo make install`
  - geht mit fast jeder aktuelleren Distribution
  - Host kann x86_64- oder IA32-basiert sein
  - „headless management" möglich (SSH Login statt KDE/GNOME Bloat)
  - Gäste kann man skriptbasiert steuern
- unterstützt Hardware-Virtualisierung (Intel VT-x & AMD-V)
- Linux® ist Hypervisor
- ¡keine Probleme mit Zeitsynchronisation!
- keine Notwendigkeit Tools im Gast zu installieren

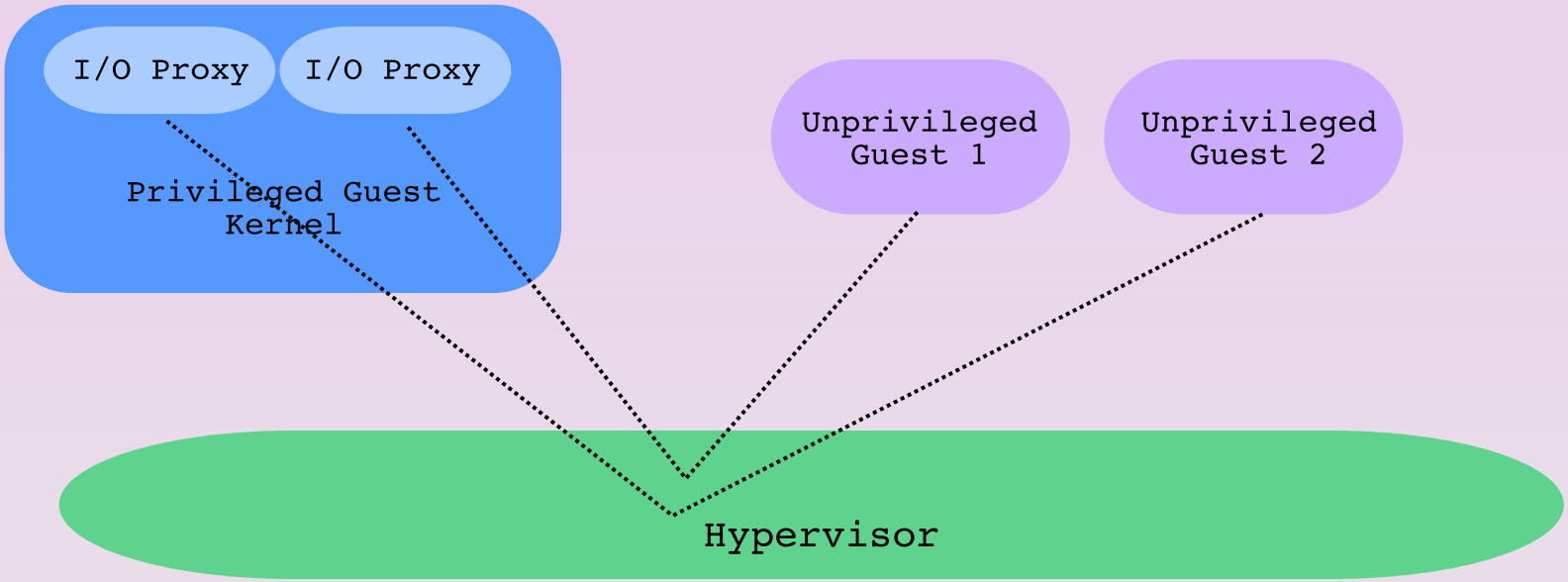
Warum KVM?

- 100% Freie Software (trotz enthaltener Innovation)
- absolut schmerzfreies Installieren und Verwalten
  - `./configure && make && sudo make install`
  - geht mit fast jeder aktuelleren Distribution
  - Host kann x86_64- oder IA32-basiert sein
  - „headless management” möglich (SSH Login statt KDE/GNOME Bloat)
  - Gäste kann man skriptbasiert steuern
- unterstützt Hardware-Virtualisierung (Intel VT-x & AMD-V)
- Linux® ist Hypervisor
- ¡keine Probleme mit Zeitsynchronisation!
- keine Notwendigkeit Tools im Gast zu installieren
- sehr performant

Warum KVM?

- 100% Freie Software (trotz enthaltener Innovation)
- absolut schmerzfreies Installieren und Verwalten
  - `./configure && make && sudo make install`
  - geht mit fast jeder aktuelleren Distribution
  - Host kann x86_64- oder IA32-basiert sein
  - „headless management" möglich (SSH Login statt KDE/GNOME Bloat)
  - Gäste kann man skriptbasiert steuern
- unterstützt Hardware-Virtualisierung (Intel VT-x & AMD-V)
- Linux® ist Hypervisor
- ¡keine Probleme mit Zeitsynchronisation!
- keine Notwendigkeit Tools im Gast zu installieren
- sehr performant
- reichhaltige Auswahl an Gastbetriebssystemen: GNU/Linux, *BSD, MS Windows XP/2000/Vista/7, Plan9

Funktionsweise - gängige Hypervisorarchitektur



Gängige Hypervisorarchitektur - Ein-/Ausgabe

Gängige Hypervisorarchitektur - Ein-/Ausgabe

1. Gast führt Ein-/Ausgabe Instruktion aus

Gängige Hypervisorarchitektur - Ein-/Ausgabe

1. Gast führt Ein-/Ausgabe Instruktion aus
2. Hypervisor fängt Instruktion ab
   1. Weiterleitung an privilegierten Gast
   2. Wechsel in privilegierten Gast

Gängige Hypervisorarchitektur - Ein-/Ausgabe

1. Gast führt Ein-/Ausgabe Instruktion aus
2. Hypervisor fängt Instruktion ab
   1. Weiterleitung an privilegierten Gast
   2. Wechsel in privilegierten Gast
3. Privilegierter Gast führt Interrupt aus
   1. Kontextwechsel in Interrupt Handler (über den Scheduler des privilegierten Gasts)
   2. E/A Prozeß führt E/A für unprivilegierten Gast aus
   3. E/A Prozeß signalisiert Hypervisor Status (über privilegierten Gast)

Gängige Hypervisorarchitektur - Ein-/Ausgabe

1. Gast führt Ein-/Ausgabe Instruktion aus
2. Hypervisor fängt Instruktion ab
   1. Weiterleitung an privilegierten Gast
   2. Wechsel in privilegierten Gast
3. Privilegierter Gast führt Interrupt aus
   1. Kontextwechsel in Interrupt Handler (über den Scheduler des privilegierten Gasts)
   2. E/A Prozeß führt E/A für unprivilegierten Gast aus
   3. E/A Prozeß signalisiert Hypervisor Status (über privilegierten Gast)
4. Hypervisor wechselt zu unprivilegiertem Gast

Gängige Hypervisorarchitektur - Ein-/Ausgabe

1. Gast führt Ein-/Ausgabe Instruktion aus
2. Hypervisor fängt Instruktion ab
   1. Weiterleitung an privilegierten Gast
   2. Wechsel in privilegierten Gast
3. Privilegierter Gast führt Interrupt aus
   1. Kontextwechsel in Interrupt Handler (über den Scheduler des privilegierten Gasts)
   2. E/A Prozeß führt E/A für unprivilegierten Gast aus
   3. E/A Prozeß signalisiert Hypervisor Status (über privilegierten Gast)
4. Hypervisor wechselt zu unprivilegiertem Gast
5. Hypervisor führt Gast Code weiter aus

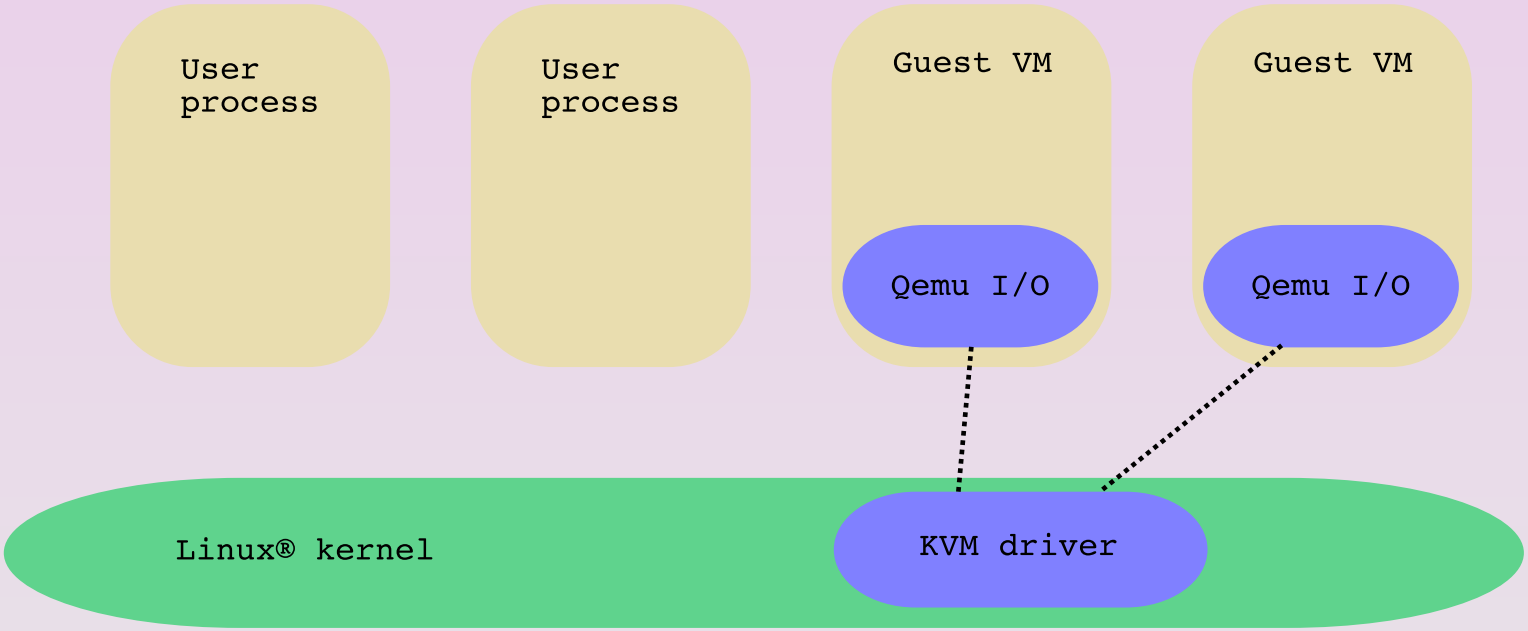
Gängige Hypervisorarchitektur - Ein-/Ausgabe

1. Gast führt Ein-/Ausgabe Instruktion aus
2. Hypervisor fängt Instruktion ab
   1. Weiterleitung an privilegierten Gast
   2. Wechsel in privilegierten Gast
3. Privilegierter Gast führt Interrupt aus
   1. Kontextwechsel in Interrupt Handler (über den Scheduler des privilegierten Gasts)
   2. E/A Prozeß führt E/A für unprivilegierten Gast aus
   3. E/A Prozeß signalisiert Hypervisor Status (über privilegierten Gast)
4. Hypervisor wechselt zu unprivilegiertem Gast
5. Hypervisor führt Gast Code weiter aus

Geht das nicht besser?

.

KVMs Hypervisorarchitektur



Ein-/Ausgabe mit KVM

Ein-/Ausgabe mit KVM

1. Gast VM führt Ein-/Ausgabe Instruktion aus

Ein-/Ausgabe mit KVM

1. Gast VM führt Ein-/Ausgabe Instruktion aus
2. Hostkern=Hypervisor fängt Instruktion ab und wechselt in VM Userspace

Ein-/Ausgabe mit KVM

1. Gast VM führt Ein-/Ausgabe Instruktion aus
2. Hostkern=Hypervisor fängt Instruktion ab und wechselt in VM Userspace
3. VM Userspace führt Ein-/Ausgabe für den Gast aus

Ein-/Ausgabe mit KVM

1. Gast VM führt Ein-/Ausgabe Instruktion aus
2. Hostkern=Hypervisor fängt Instruktion ab und wechselt in VM Userspace
3. VM Userspace führt Ein-/Ausgabe für den Gast aus
4. VM Userspace kehrt in Kernelspace zurück

Ein-/Ausgabe mit KVM

1. Gast VM führt Ein-/Ausgabe Instruktion aus
2. Hostkern=Hypervisor fängt Instruktion ab und wechselt in VM Userspace
3. VM Userspace führt Ein-/Ausgabe für den Gast aus
4. VM Userspace kehrt in Kernelspace zurück
5. Hostkern=Hypervisor setzt Gast Code fort

Klar geht es besser!

.

Linux® als Hypervisor

Linux® als Hypervisor

- KVM „mißbraucht” Linuxkern als Hypervisor
  - Hostsystem ist minimale GNU/Linux Installation
  - KVM ist Teil des Kerns
  - Tools zur Verwaltung installiert

Linux® als Hypervisor

- KVM „mißbraucht” Linuxkern als Hypervisor
  - Hostsystem ist minimale GNU/Linux Installation
  - KVM ist Teil des Kerns
  - Tools zur Verwaltung installiert
- Vorteile
  - Host ist ein Server mit GNU/Linux System
  - breite Unterstützung von Hardware
  - keine / kaum zusätzliche Software nötig
  - kommt „ohne” Management Tools aus

Linux® als Hypervisor

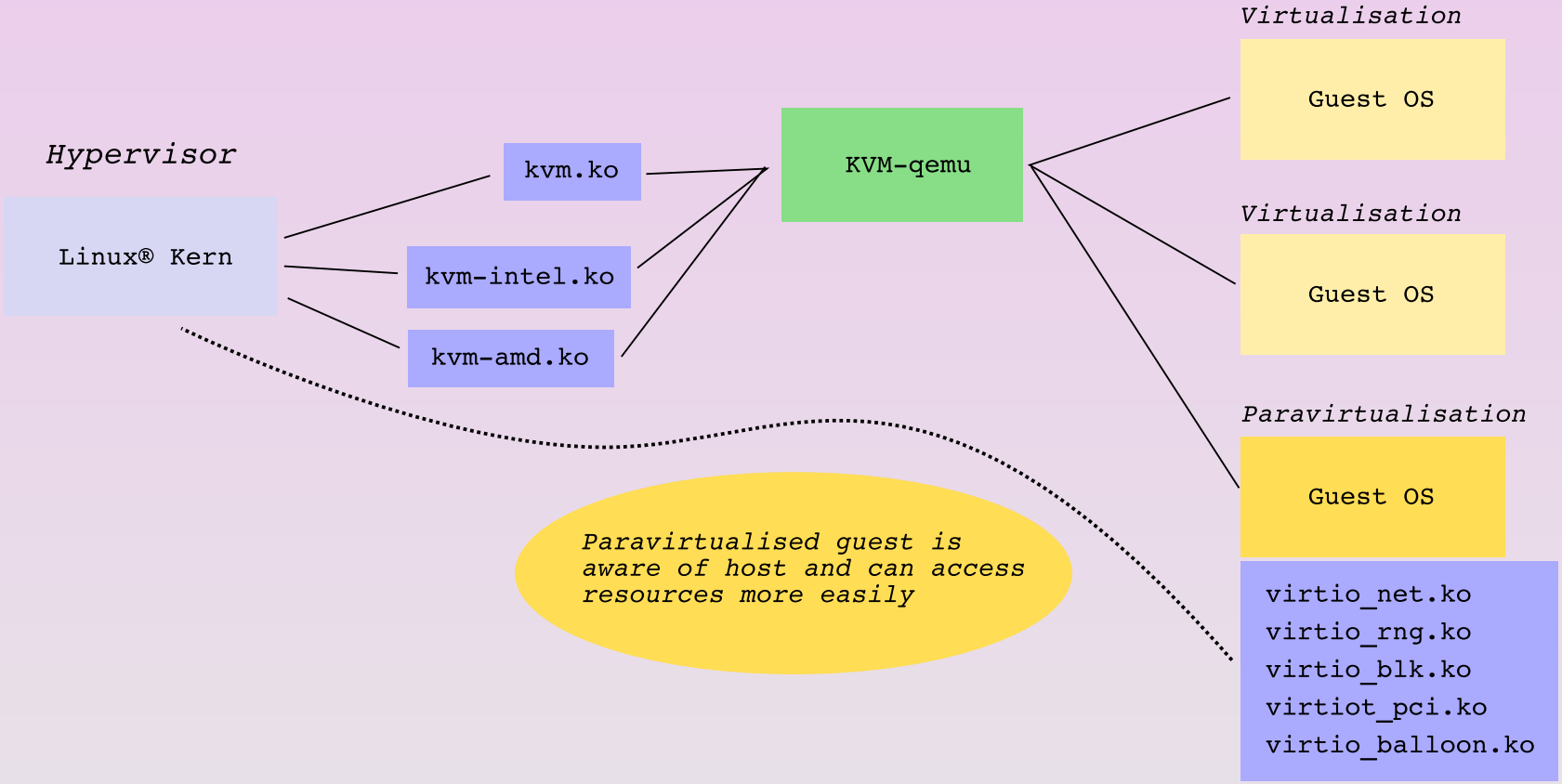
- KVM „mißbraucht” Linuxkern als Hypervisor
  - Hostsystem ist minimale GNU/Linux Installation
  - KVM ist Teil des Kerns
  - Tools zur Verwaltung installiert
- Vorteile
  - Host ist ein Server mit GNU/Linux System
  - breite Unterstützung von Hardware
  - keine / kaum zusätzliche Software nötig
  - kommt „ohne” Management Tools aus
- Nachteile
  - Host ist ein Server mit GNU/Linux System
    $\longrightarrow$ komplexer als reiner Hypervisor
  - KVM ist noch junges Projekt

KVM - Komponenten

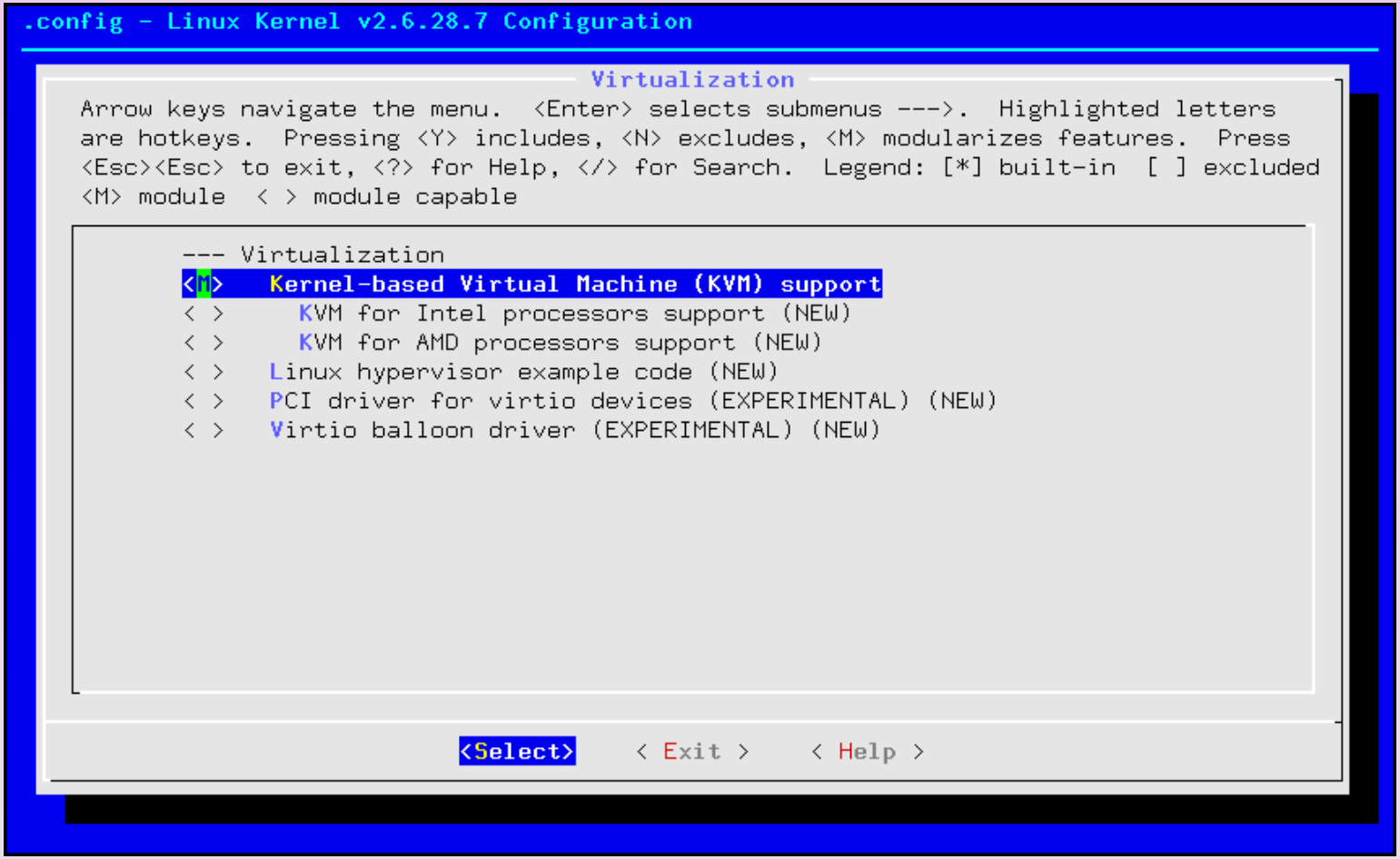
KVM - Komponenten

- KVM Kernelmodul (GPL v2)
- KVM Benutzermodul (LGPL v2)
- QEMU Virtual CPU Library (`libqemu.a`, LGPL)
- QEMU PC Emulator (LGPL)
- modifizierter Linux® QEMU Emulator (GPL)
  - verwaltet Gast Adreßraum über `/dev/kvm`
  - verwaltet E/A und Anzeige
- BIOS Dateien (`bios.bin`, `vgabios.bin`, `vgabios-cirrus.bin`)

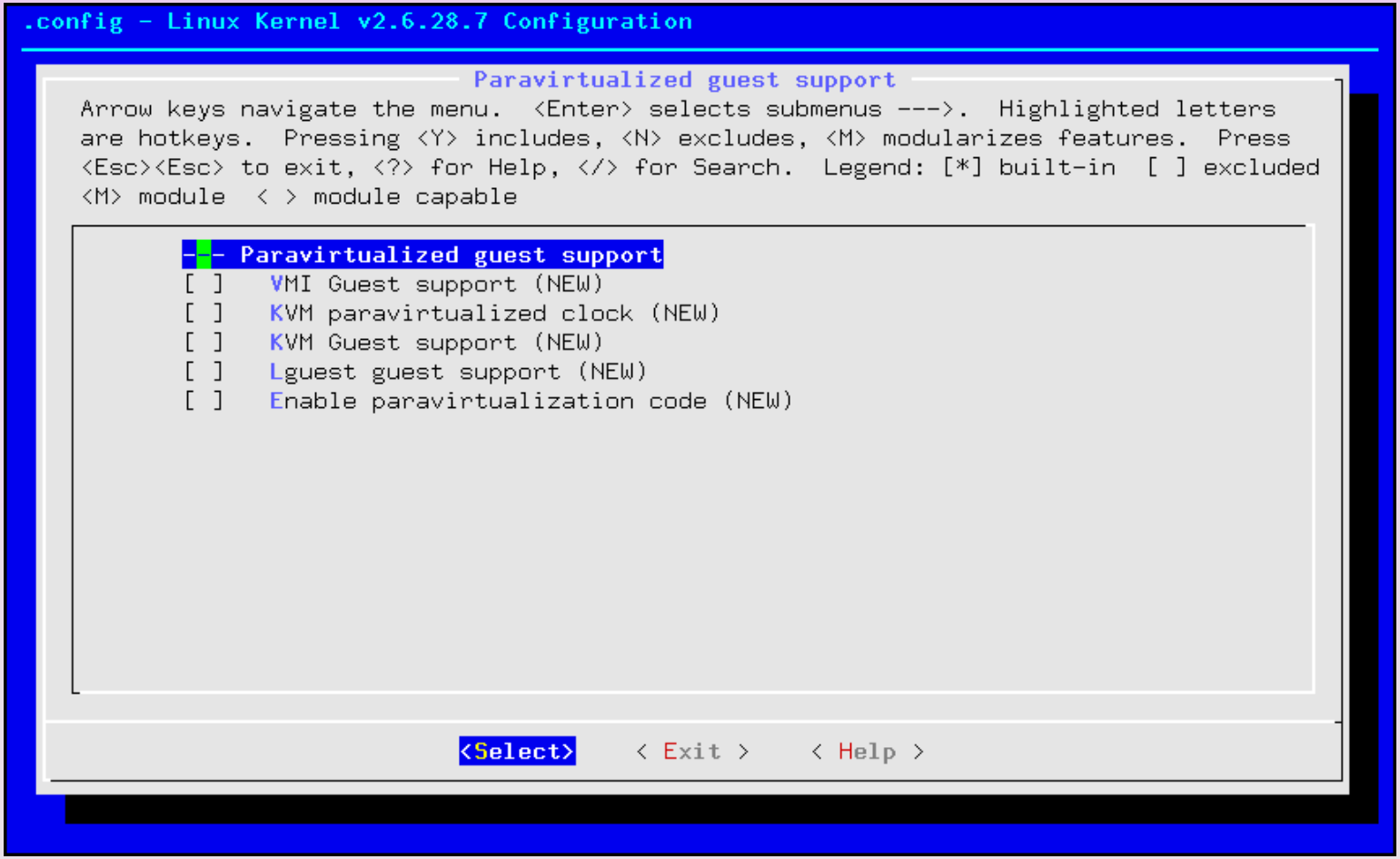
KVM - Übersicht



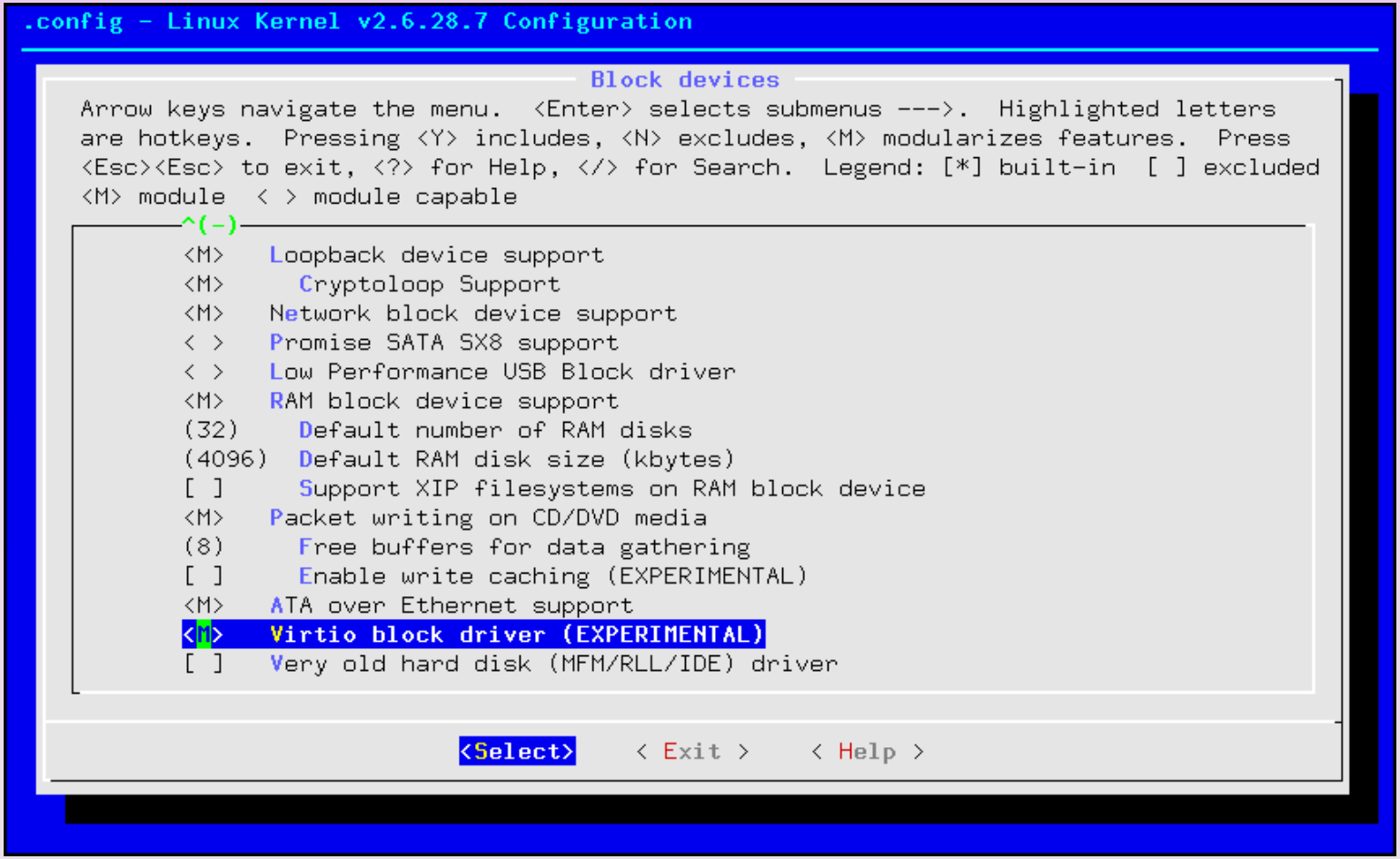
KVM Linuxkern - Hypervisor Kerneloptionen



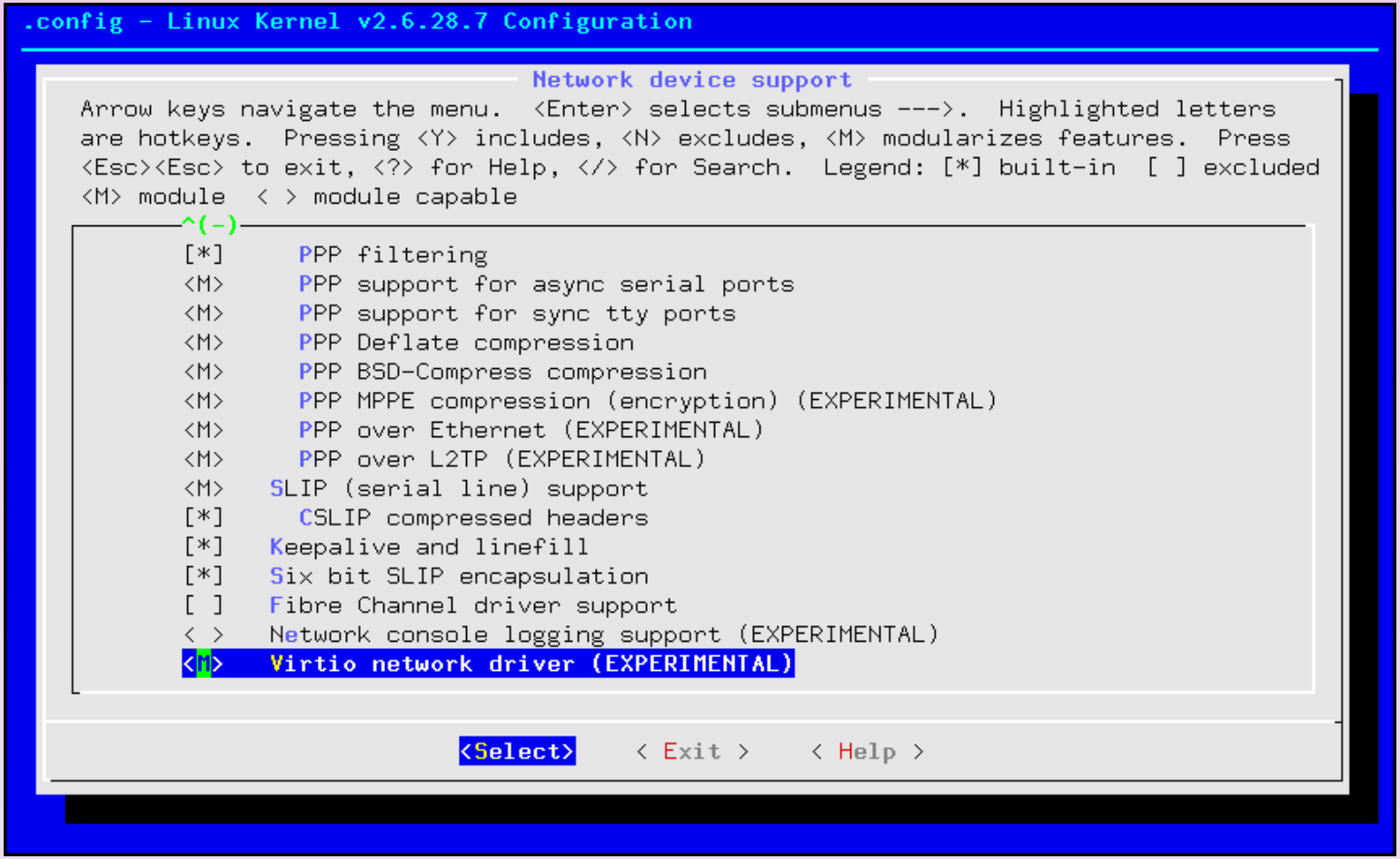
KVM Gastkern - Virtualisierungsoptionen



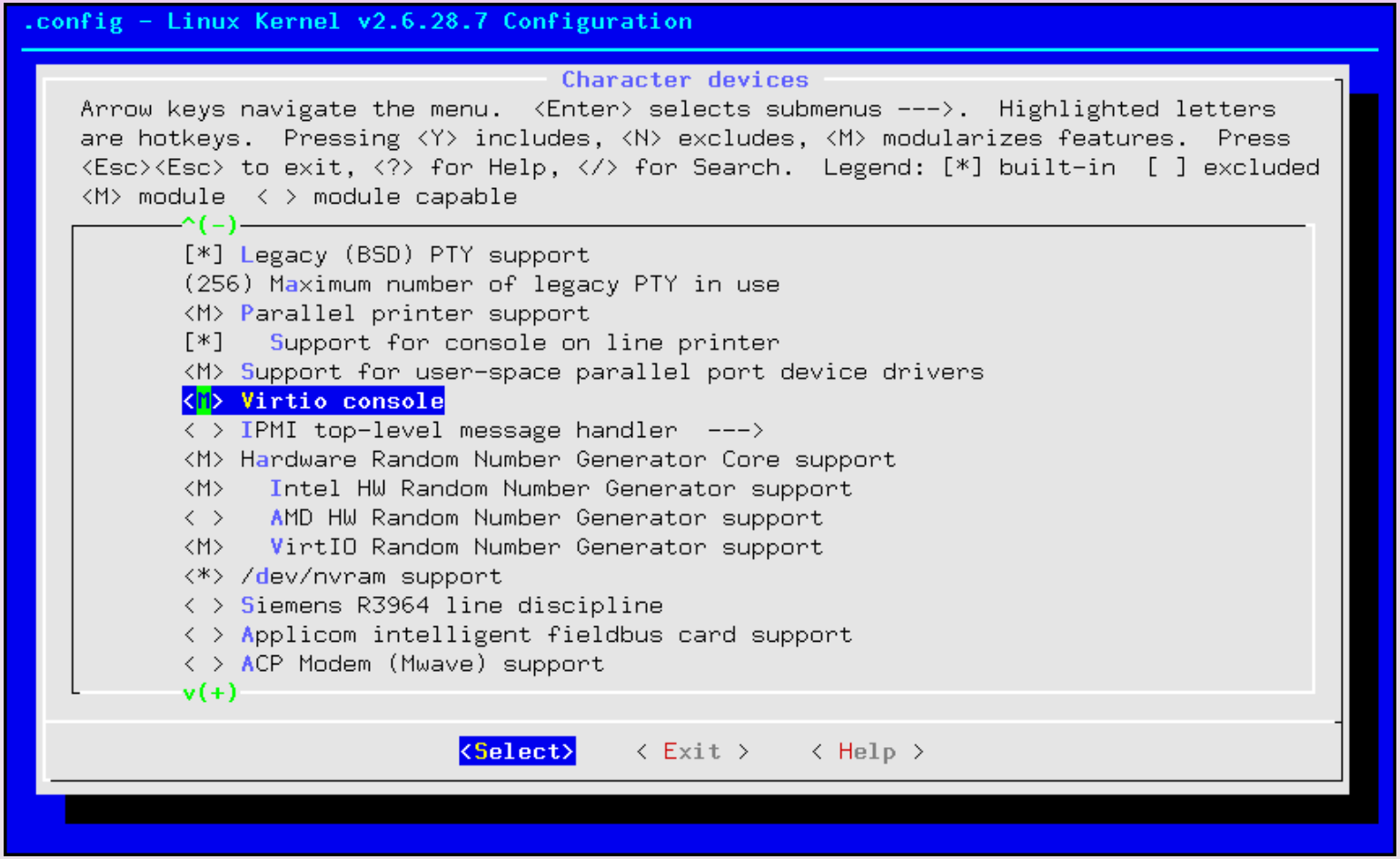
KVM Gastkern - Optionen für Blockgeräte



KVM Gastkern - Optionen für Netzwerkgeräte



KVM Gastkern - Optionen für Verschiedenes



Disk Images vorbereiten

Disk Images vorbereiten

- `qemu-img` verwaltet Disk Image Dateien
  - Disk Image Dateien können *sparse* sein

Disk Images vorbereiten

- `qemu-img` verwaltet Disk Image Dateien
  - Disk Image Dateien können *sparse* sein
- `qemu-img` versteht mehrere Formate
  - Network Block Device (NBD)
  - Parallels, `qcow2`/`qcow`, `vpc`, `bochs`, `dmg`, `vmdk`
  - Loopback Dateisysteme
  - Virtual FAT (Verzeichnisbäume als FAT nur-lesen)
  - Raw Devices

Disk Images vorbereiten

- `qemu-img` verwaltet Disk Image Dateien
  - Disk Image Dateien können *sparse* sein
- `qemu-img` versteht mehrere Formate
  - Network Block Device (NBD)
  - Parallels, `qcow2`/`qcow`, `vpc`, `bochs`, `dmg`, `vmdk`
  - Loopback Dateisysteme
  - Virtual FAT (Verzeichnisbäume als FAT nur-lesen)
  - Raw Devices
- `qemu-img` kann zwischen Formaten konvertieren

Disk Images vorbereiten

- `qemu-img` verwaltet Disk Image Dateien
  - Disk Image Dateien können *sparse* sein
- `qemu-img` versteht mehrere Formate
  - Network Block Device (NBD)
  - Parallels, `qcow2`/`qcow`, `vpc`, `bochs`, `dmg`, `vmdk`
  - Loopback Dateisysteme
  - Virtual FAT (Verzeichnisbäume als FAT nur-lesen)
  - Raw Devices
- `qemu-img` kann zwischen Formaten konvertieren
- verschlüsselte Images möglich

Gäste starten

Gäste starten

- `qemu-system-`*arch* (z.B. `qemu-system-x86_64`)

Gäste starten

- `qemu-system-`*arch* (z.B. `qemu-system-x86_64`)
- Gastanzeige über
  - SDL direkt am Bildschirm (lokaler/entfernter X Server)
  - VNC
  - VNC mit SSL/TLS Verschlüsselung

Gäste starten

- `qemu-system-`*arch* (z.B. `qemu-system-x86_64`)
- Gastanzeige über
  - SDL direkt am Bildschirm (lokaler/entfernter X Server)
  - VNC
  - VNC mit SSL/TLS Verschlüsselung
- serielle Konsole / Telnetzugang zur Verwaltung möglich

Gäste starten

- `qemu-system-`*arch* (z.B. `qemu-system-x86_64`)
- Gastanzeige über
  - SDL direkt am Bildschirm (lokaler/entfernter X Server)
  - VNC
  - VNC mit SSL/TLS Verschlüsselung
- serielle Konsole / Telnetzugang zur Verwaltung möglich
- Angabe von Disks und CD-ROMs beim Start

Gäste starten

- `qemu-system-`*arch* (z.B. `qemu-system-x86_64`)
- Gastanzeige über
  - SDL direkt am Bildschirm (lokaler/entfernter X Server)
  - VNC
  - VNC mit SSL/TLS Verschlüsselung
- serielle Konsole / Telnetzugang zur Verwaltung möglich
- Angabe von Disks und CD-ROMs beim Start
- Angabe von Netzwerkkarten beim Start

Gäste starten

- `qemu-system-`*arch* (z.B. `qemu-system-x86_64`)
- Gastanzeige über
  - SDL direkt am Bildschirm (lokaler/entfernter X Server)
  - VNC
  - VNC mit SSL/TLS Verschlüsselung
- serielle Konsole / Telnetzugang zur Verwaltung möglich
- Angabe von Disks und CD-ROMs beim Start
- Angabe von Netzwerkkarten beim Start
- Wahl des Speichers, der CPUs und Optionen

Gastnetzwerk

Gastnetzwerk

- User Networking
  - Parameter `-net nic -net user`
  - Gast verwendet internen DHCP Server
  - MAC Adresse wird automatisch generiert

Gastnetzwerk

- User Networking
  - Parameter `-net nic -net user`
  - Gast verwendet internen DHCP Server
  - MAC Adresse wird automatisch generiert
- privates Netzwerk zwischen KVM Gästen
  - lokale Bridge Interface auf KVM Host
  - Skript `qemu-ifup` zum Einbinden der Gastnetzwerkkarte
  - `-net nic,macaddr=DE:AD:BE:EF:19:84 -net tap`

Gastnetzwerk

- User Networking
  - Parameter `-net nic -net user`
  - Gast verwendet internen DHCP Server
  - MAC Adresse wird automatisch generiert
- privates Netzwerk zwischen KVM Gästen
  - lokale Bridge Interface auf KVM Host
  - Skript `qemu-ifup` zum Einbinden der Gastnetzwerkkarte
  - `-net nic,macaddr=DE:AD:BE:EF:19:84 -net tap`
- öffentliches Netzwerk zwischen KVM Gästen
  - lokale Bridge mit einem externen Netzwerkgerät
  - sonst wie privates Netzwerk zwischen KVM Gästen

Gäste starten - Beispiel

Gäste starten - Beispiel

```
sudo qemu-system-x86_64 \
 -drive file=/srv/kvm/grmltest.qcow2,if=virtio,boot=off \
 -cdrom /nfs/ISO/grml/grml_2008.11.iso \
 -smp 2 -m 2048 \
 -boot d \
 -daemonize \
 -name Schluchtenscheisser \
 -net nic,macaddr=DE:AD:BE:EF:19:84,model=virtio \
 -net tap,script=/etc/qemu-ifup \
 -vnc 10.2.3.4:1 -k de \
 > /tmp/grml.log 2>&1
```

Gäste verwalten

Gäste verwalten

- Verwaltung von Gast direkt
- Grafische Konsole in SDL/X Anzeige
- Telnet Konsole
- Libvirt Tools
  - API für QEMU, KVM und Xen™
  - einheitlicher Zugriff auf Gäste
  - Management Tools (z.B. Red Hat Virtual Machine Manager
- SSH / Telnet (für *headless mode*)

Noch Fragen?

Über dieses Dokument

- Autor: René „Lynx” Pfeiffer
- Erstellt mit LaTeX und LaTeX Beamer Class
- Dokumentensammlung unter
  http://web.luchs.at/information/docs.php