

QEMU and Kernel-based Virtual Machine - Wiki

(Link zu dieser Seite als [[QEMU-KVM-Book/ Content]])

[Order \(print version\)](#) | [Recommendation](#) | [The Authors](#) | [Imprint, Contact](#)

Introduction

Basics

Installation

Quick Start

QEMU-Monitor

Storage Media

Virtual Hardware

Network Options

Special Options

Management Tools

Host-Systems

 QEMU and KQEMU under Linux

 QEMU ist running not only on Linux.

 QEMU under Microsoft Windows and Wine

 QEMU under OS/2 Warp 4 and eComstation

 QEMU under DOS

 QEMU under Mac OS X

 QEMU under BSD (Solaris, FreeBSD, NetBSD, OpenBSD)

Guest Systems

Appendix

Archive of the old unofficial Wiki of QEMU

[Order \(print version\)](#) | [Recommendation](#) | [The Authors](#) | [Imprint, Contact](#)

Von „http://qemu-buch.de/de/index.php/QEMU-KVM-Book/_Content“

Diese Seite wurde bisher 50.097 mal abgerufen. Diese Seite wurde zuletzt am 30. Juli 2010 um 06:08 Uhr geändert. Inhalt ist verfügbar unter der GNU Free Documentation License 1.2.



Emulation virtualization computer literature training materials

(Link to this page as [[QEMU-KVM-Buch / General / Introduction]])

| # # # | >>> | English

Introduction

Virtual machines provide defined system environments. For templates with a specific version of an operating system and a desired patch level virtual machines are generated. Snapshots save states of a virtual machine. When an unwanted change can be restored to their former condition. Virtualization and emulation allow the independence of the hardware. Thus, for example, to run old versions of an application to develop new hardware or software for non-existing hardware. These are the strengths of QEMU. QEMU is free and runs on many systems (Linux, BSD, Mac OS X, Microsoft Windows, eComStation or DOS).

Virtualization brings a number of advantages, depending on the type are weighted differently. Given that several guest systems can run on a parallel computer, increases hardware utilization is possible.

Previously, for critical applications used in each case a physical computer. The result is a large machine park with a correspondingly high cost (acquisition, maintenance, power, air conditioning). If, however, individual applications running in a virtual machine, the applications are decoupled from each other and the hardware is better utilized. Besides security, the availability is increased, because virtual machines can be transferred in case of hardware problems on a different hardware. This optimization of the data center through virtualization solutions are called server consolidation. The Kernel-based Virtual Machine (KVM) is an appropriate virtualization solution. KVM is open source and runs on Linux.

The management of virtual machines is done under different virtualization solutions is very different. In a heterogeneous data center, it is often impossible to control with a tool more virtualization solutions. To solve this problem, the C library is designed libvirt. It provides standard interfaces for managing different virtualization solutions.

The print edition

QEMU, the Kernel-based Virtual Machine (KVM) and the C library in the libvirt wiki <http://qemu-buch.de> described in detail. All text is available under the GNU Free Documentation License. The lyrics of this Wiki have been adjusted for this book. This book is both a workbook and a reference book. It is aimed at (Linux) system administrators, software developers, software testers, and students interested in technology. The many examples of applications and the license used, it is particularly suitable as training material.

It uses the following notations.

Courier font

For console commands and source code.

~ # **Command**

For Unix console commands as system administrator (root).

~ \$ **Command**

For Unix console commands as normal user.

Host ~ # **command**

For Unix console commands as system administrator (root) on the host computer.

~ \$ Host **command**

For Unix console commands as normal user on the host computer.

```
Host C: \> command
```

For DOS / Windows console commands on the host computer. As the options regardless of the operating system, here is usually the spelling for the Unix console uses.

```
Host ~ # command
```

For Unix console commands as system administrator (root) on the host system.

```
Guest ~ $ command
```

For Unix console commands as a normal user on the host system.

```
Guest C: \> command
```

For DOS / Windows console commands on the host computer.

```
(Qemu) command
```

For orders of the QEMU monitor.

```
virsh # command
```

For instructions of the program *virsh* (*libvirt* library).

```
Host A long ~ $ command, \
the line can not be listed in a.
```

If a command can not be represented in a line, the line break is represented by a backslash. The command is entered in one line.

In http://qemu-buch.de/d/Anhang/_Nützliche_Tools_KVM are some useful tools for working with QEMU and explained. This overview also serves to get along with less well-known guest systems. Most of the described tools are easy to use as a command-line commands, and come from the Unix environment. They are also on other platforms such as Microsoft Windows released.

```
| # # | >>> http://qemu-buch.de
```

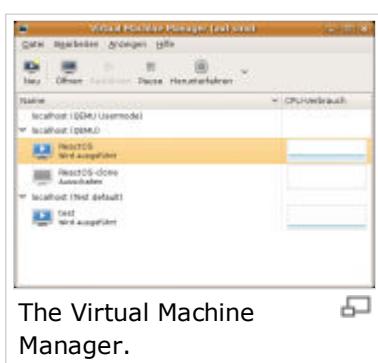
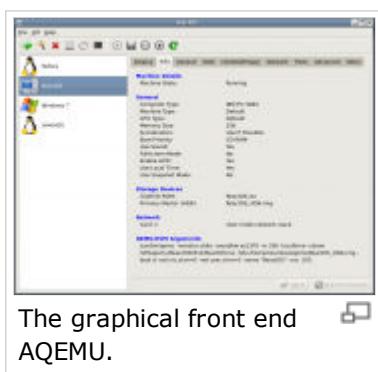
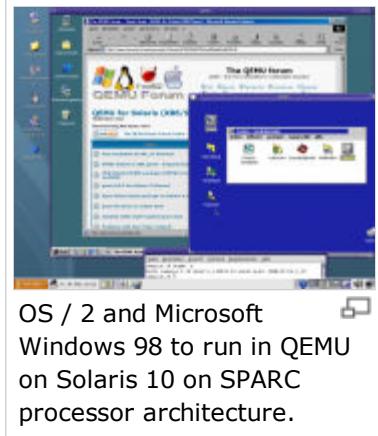
Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Allgemeines/_Einleitung "

This page has been accessed 25,187 times. This page was last updated on 23 March 2010 at 14:11 clock changed. Content is available under GNU Free Documentation License 1.2 .

QEMU-KVM libvirt virtualization hardware emulation Native virtualization, paravirtualization, full virtualization, hypervisor, Single Kernel Image

(Link to this page as [[QEMU-KVM-Buch / basis]])

<<< | # # # | >>> | English



Contents

1 Basics

- 1.1 Virtualization
 - 1.1.1 Hardware Emulation
 - 1.1.2 Native Virtualization
 - 1.1.3 paravirtualization / Full Virtualization
 - 1.1.4 operating system level virtualization
- 2.1 Emulation
 - 1.2.1 Hardware Emulation
 - 1.2.2 API emulation
- 1.3 The versatile QEMU
- 1.4 The Kernel-based Virtual Machine on the rise
- 1.5 The libvirt library



The Qemu Manager for Windows on Mac OS X.
WineBottle

Virtualization

The primary goal of virtualization is the decoupling of software from the available hardware resources to allocate them optimally to the isolated individual systems. It is all about the core components of CPU, memory, disk space and network connectivity. Virtualization allows the placement of specific virtual machines, each containing only the necessary services for their task. Thus, for although on a physical machine both a Web server, a mail server and an FTP server running simultaneously, but for security reasons it is not recommended. If one of the three services is compromised, the entire machine is affected. However, these services are in each virtual machine isolated from each other on the system operated, only the affected virtual machine to be replaced. Another advantage is the server consolidation. The virtual machines are assigned exactly the resources needed for a particular task. If the requirements will be modified as resources without changing the hardware.

Hardware Emulation

In hardware emulation, the entire hardware of a computer is simulated. It is possible to install and run on operating systems that are designed for other processors. This is simulated including the full instruction set of the host processor via the host processor. The emulation but has a cost. Since each CPU instruction of the host system CPU into corresponding commands on the host system must be translated, the execution speed of programs is low. Another disadvantage is the lack of a dynamic resource management. That is, during the term can host systems resources are not allocated variable. Without QEMU accelerator is a hardware emulator.

Native Virtualization

The Native Virtualization, also known as hardware virtualization, the guest operating system parts of the physical hardware is in the form of virtual hardware. To ensure that only guests are served that are compatible with the CPU of the host. The privileged CPU instructions, the guest systems are directly managed by the kernel while the host system. In contrast hits on devices (video card, network card, etc.) made via emulated standard components. Known members of this virtualization approach is VirtualBox, VMware Player, VMware Server, VMware Workstation and Parallels Desktop / Workstation. The execution speed of the guest systems, this method is high. QEMU supports up to version 0.11 the optional accelerator KQEMU and thus the Native Virtualization.

paravirtualization / Full Virtualization

In contrast to hardware emulation and the Native Virtualization offers paravirtualization only similar interface as the physical hardware. The critical CPU commands that directly address the hardware, must first be rewritten in the host system and therefore need not be intercepted and replaced by more complicated procedures. A sleek program, the hypervisor (virtual machine monitor), does the scheduling



Warnke, Ritzau
qemu-kvm and libvirt
4. Edition 2010
ISBN: 978-3-8370-0876-0
276 pages, 27.27 EUR
Order

tasks for the resources of virtual machines. The disadvantage is that the guest operating systems must be adapted. Only processors with hardware virtualization technology (Intel's Vanderpool, AMD Pacifica) enable you to run unmodified guest systems. This is called the Full Virtualization, or the hardware-assisted virtualization. The latter is not to be confused with the hardware emulation. One advantage of paravirtualization and full virtualization is the much higher execution speed. Another advantage is the live migration, which is transmitted to the virtual machines on the fly from a host system to another. The host systems are continuously available, even if one of the hosts are shut down for maintenance work. There are two types of hypervisor:

Type 1 hypervisor: The hypervisor runs directly on the hardware and provides the guest systems, the resources available. Typical examples are Xen (open source) and VMware ESX (i). With Xen and VMware ESX Hypervisor is above a privileged virtual machine (Linux) to control.

Type 2 hypervisor: The hypervisor runs on an operating system that the I / O resources provide. A typical example of the Kernel-based Virtual Machine (KVM) is. KVM supports full virtualization. Further support VirtualBox, VMware Player, VMware Workstation and VMware Server Full Virtualization.

OS-level virtualization

When virtualization at the OS level, divide the host and guest systems with kernel (single kernel image - SKI). Here, the virtualized operating system and not the hardware. The host operating system generates more instances of itself this reason only guest systems with the same kernel as the host system supported. Since the overhead is low, the execution speed is high. Typical representatives of this category, the Solaris Zones, BSD jails, Linux Containers (LXC) are OpenVZ and Linux-VServer. QEMU does not have this type of KVM virtualization assigned.

Emulation

In connection with virtualization is often called the term emulation. As emulation is referred to in computing the functional simulating a system by another. Recreate the system receives the same data, executes the same programs and achieve the same results as the original system. An emulator is a system that imitates another. the emulated system is operated isolated present, it is not a virtualization.

Hardware Emulation

In the section *virtualization* has already described the hardware emulation. Since the emulated computers are isolated from each host system, this type of emulation virtualization. The hardware emulation is not to be confused with the hardware-assisted virtualization. QEMU without acceleration, is described as a hardware emulator. KVM can not be assigned to the hardware emulation.

API emulation

and libraries of another operating system modeled - In this type of emulation interfaces (API Application Programming Interface). This can run programs of the other operating system. This type of emulation is not necessarily isolated to running programs. They run along with emulation libraries directly in the host system. This is not a virtualization. A typical example is Cygwin. Cygwin on Windows provides the API functions of Linux. Under Cygwin you have full access to the host system. Cygwin is therefore not a virtualization solution. The versatile QEMU under Linux, BSD and Mac OS X / Darwin also an OS API emulation. The user space emulation allows execution of programs (binaries) that are compiled for other libraries.

The user space emulation QEMU is used by the software Darwine under the PowerPC architecture. Darwin, now WineBottle is a port of Wine (Wine Is Not an Emulator, see http://qemu-buch.de/d/Anhang/_Nützliche_Tools/_Wine) on Mac OS X. The goal is to run Windows applications on Mac OS X.

The versatile QEMU

Websites: <http://wiki.qemu.org>

The free, open-source QEMU emulates the entire hardware of a computer's CPU (hardware emulation). QEMU is not, such as VMware, on the x86 architecture is limited. QEMU achieved by the Dynamic translation is a good speed. QEMU accelerator 0.11.x to the optional KQEMU was supported (Native Virtualization). The development of KQEMU has been set. 12:12 From QEMU building on the advantages

of the Kernel-based Virtual Machine (see below).

QEMU runs on many operating systems (Linux, BSD, Mac OS X, Microsoft Windows, eComStation, DOS) and processor architectures and is easy to install. To install QEMU accelerator without any administrator privileges. Thus, a virtual machine, including QEMU be burned to a DVD and started on another computer.

QEMU allows you to save states of the virtual machine several times. These VM snapshots, changes to the system can be reversed. QEMU has features not found in other virtualization software. For example, QEMU supports live migration, debugging the system and boot from older disk formats. It can emulate hardware failure. It is also possible to set the system time of the host system as desired. Live-CD/DVDs-and boot floppy to start with QEMU, without restarting the computer.

For QEMU package includes the powerful tool *qemu-img*. It is used to create, convert, and encrypt an image file (virtual hard disks) in various formats. QEMU can boot a virtual machine image files directly from other virtualization software. Using QEMU, you can move virtual machines for the Kernel-based Virtual Machine. It is thus possible in hardware on the Kernel-based Virtual Machine is not running. Furthermore, part of the package, the tool QEMU *qemu-nbd*. It is used to export of image files over the network using the NDB Protocol (Network Block Device).

Under Linux, BSD and Mac OS X QEMU also supports user-space emulation. This API emulation that allows executable programs compiled for other dynamic libraries, operate in user space. Here, the processors x86, PowerPC, ARM, MIPS 32-bit, Sparc32/64, ColdFire (m68k), ETRAX CRIS support and MicroBlaze.

QEMU is developed under Linux and open source software. There are ports for many operating systems. Many other virtualization (KVM, Sun xVM VirtualBox, Xen, FAUmachine, Win4BSD, Win4Solaris, Win4Lin, and Darwin) use parts of the source code of QEMU. In return, the source code of these products will be re-integrated into QEMU. Thanks to the openness of the source code and configuration options start with the development of additional tools is problematic.

Kernel-based Virtual Machine on the rise

Websites: <http://www.linux-kvm.org> <http://www.linux-kvm.com>

The Kernel-based Virtual Machine (KVM) is considered to whiz in the virtualization solutions. One reason for this is that KVM consistently on the benefits of processors with hardware virtualization technology (AMD Pacifica, Intel's Vanderpool) sets. The Kernel-based Virtual Machine that is used in the Full Virtualization (type 2 hypervisor). KVM support usually is not the paravirtualization. There is a patch that enables the paravirtualization of Linux guests. This patch was not included in the Linux kernel. The use of paravirtualized device drivers for guest systems, however, is possible. The advantage of this driver is the lower overhead and higher performance.

KVM supports other CPU extensions such as page tables (Nested Page Tables - AMD Extended Paging Tables - Intel) and IOMMU (I / O Memory Mapping Unit). Page tables are used for translation of guest in host addresses. Thus, the hypervisor is the task of each address request from a host system relieves a shadow page table to be determined.

For KVM, the Kernel Merging Same Page (KSM) in the Linux kernel has been integrated as of version 2.6.32. KSM consolidates identical memory areas. This is more memory to virtual machines made available as a physical memory available.

KVM and this CPU extensions enable the Nested Virtualization. That is, in a virtual environment in turn virtualization solutions. A Kernel-based Virtual Machine can therefore under a Kernel-based Virtual Machine to be started and contain virtual machines.

Components of the KVM kernel modules *kvm.ko*, the hardware-specific modules or *kvm-kvm-intel.ko* *amd.ko* and the device file */dev/kvm*. After loading the modules of the Linux kernel itself works as a hypervisor. The KVM kernel modules prevent the execution of other virtualization solutions, such as Sun xVM VirtualBox and VMware Server, Workstation and Player. Xen is an advantage over the use of only a privileged system. Xen privileged requires two systems: the hypervisor and the special guest system *domain 0* (Linux). First, because the installation of KVM is easier than Xen, the other is a privileged system to secure more than two.

KVM is not a virtual hardware. This does QEMU. The development of KVM and QEMU is merged. Why KVM is described here together with QEMU. Many Linux distributions include QEMU and *KVM KVM*) as a

package (one of *qemu*. Unlike QEMU KVM only supports host systems with x86 processors.

KVM is open source and runs on Linux kernel version 2.6.20. KVM Qumranet is developed by Israeli companies. Qumranet has since been bought by Red Hat, the KVM said the focus of its virtualization strategy. Based on KVM, Red Hat *Red Hat Enterprise* complete virtualization solution *for Server Virtualization* developed. The operational area of the Kernel-based Virtual Machine, the Server Consolidation (Data Center Virtualization) is. However, KVM used on the desktop and replaces KQEMU.

The libvirt library

Website: <http://libvirt.org>

While many virtualization solutions are now free, appropriate management tools are usually expensive. Furthermore, in a heterogeneous data center management tools are often several different virtualization solutions are required. To solve these problems, the library developed *libvirt*. *Libvirt*, a virtualization layer between software and management tools. Building on this layer manage appropriate management tools (*Virtual Machine Manager virsh*) the different virtualization solutions. already supports QEMU, KVM, Xen, VirtualBox, VMware ESX, LXC Linux container system, OpenVZ Linux container system, and Mode Linux. *libvirt* is released as open source users. For various Linux distributions are applicable software packages. The library can be *Cygwin* and *MinGW* under Microsoft Windows to compile.

<<< | # # # | >>> <http://qemu-buch.de>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Grundlagen "

This page has been accessed 32,978 times. This page was last updated on 27 September 2010 at 17:40 clock changed. Content is available under GNU Free Documentation License 1.2 .



Translate

Translate from: German

Translate into: English

KVM QEMU download installation: qemu-kvm qemu-kvm-extras build Linux, Kernel Same Page Merging KSM

(Link to this page as [[QEMU-KVM-Buch / installation]])

<<< | # # # | >>> | English

Contents

1 Installation of QEMU

1.1 QEMU on Linux

- 1.1.1 Software packages
- 1.1.2 compile from source
 - 1.1.2.1 QEMU on Ubuntu 0.13.0
- 1.1.3 KVM kernel modules
- 1.1.4 Problems

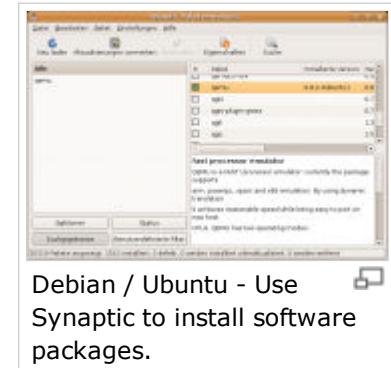


The installation of QEMU can be tested in a virtual machine.

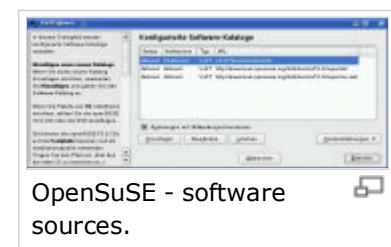
Installation of QEMU

The installation of QEMU is easy for the major operating systems and is made with the usual methods. Installation instructions can be found at each of the specified URL.

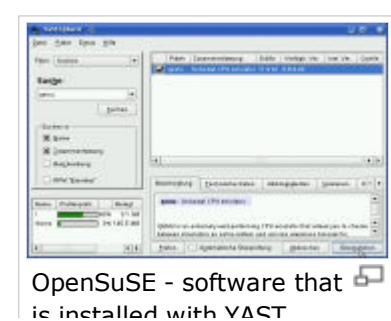
Microsoft Windows and Wine	http://qemu-buch.de/d/QEMU_unter_Microsoft_Windows
OS / 2 Warp 4 and eComStation	http://qemu-buch.de/d/QEMU_unter_eComstation
DOS	http://qemu-buch.de/d/QEMU_unter_DOS
Mac OS X (x86)	http://qemu-buch.de/d/QEMU_unter_Mac_OS_X
FreeBSD, OpenBSD, NetBSD, Solaris	http://qemu-buch.de/d/QEMU_unter_BSD



Debian / Ubuntu - Use Synaptic to install software packages.



OpenSuSE - software sources.



OpenSuSE - software that is installed with YaST

QEMU on Linux

The support for the QEMU accelerator KQEMU ends with 12:11 *. From QEMU 0.12.0, only the Kernel-based Virtual Machine (KVM) is supported. On computers without hardware virtualization technology is KVM but not used. Who would not buy any new hardware and accelerate QEMU will still need to install QEMU 00:11 .* or older. A guide can be found at the URL http://qemu-buch.de/d/QEMU+KQEMU_unter_Linux .

The support of the hardware virtualization is disabled in the BIOS usually. This prevents rootkits that use the hardware virtualization. be activated for use of the KVM kernel modules must support hardware virtualization (example):

KVM QEMU HOWTO Quick Start Manual

Documentation Documentation is qemu, kvm one, computer literature

(Link to this page as [[QEMU-KVM-Buch / Quick Start]])

<<< | # # # | >>> | English

Quick Start

When you create a virtual machine the following steps:

1. First, a virtual disk (disk image) is generated.
2. Then QEMU or KVM is started with this virtual hard disk and the installation media. The virtual machine is configured with boot options.
3. The operating system, as installed in a real machine. It is only on the support of the emulated hardware respected. Special driver must not be installed in the host system.

The QEMU boot options are on all operating systems equal. To demonstrate the key options in the example below, a virtual machine using shell commands will be generated. The call to QEMU on the command line has the advantage that error messages are displayed immediately. For this example, the operating system ReactOS (<http://www.reactos.org/de/>) selected as the host system. To install the operating system a virtual disk is created. This is done with the package contained in the QEMU *qemu-img* tool and *create* the parameters. In some distributions this tool is called *kvm-img*. It is the name and size of the virtual harddisk. With *the-f* format, the preset.

```
Host ~ $ qemu-img create-f qcow2 ReactOS.img 1G
```

From the ReactOS website is the image of the installation CD to download and unpack.

```
Host ~ $ unzip ReactOS *- iso.zip
```

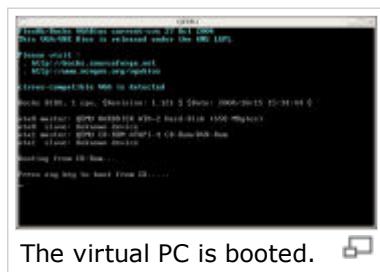
Is called QEMU, or the Kernel-based Virtual Machine normally *qemu* with the command. Some distributions call it made with the *kvm*, *qemu-system-x86_32*, *qemu-system-x86_64* *qemu* or *kvm*. To install the virtual machine is started from the installation CD. Whose name is entered after the *option-cdrom*. The name of the created virtual hard drive on the *option-typed hda*. This virtual computer from the CD will boot the virtual, the *option-boot d* used.

```
Host ~ $ qemu-hda ReactOS.img-cdrom-boot d ReactOS.iso
```

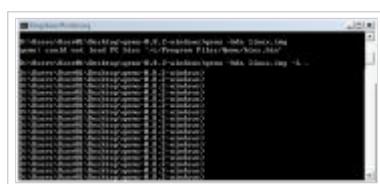
If the error message appear at startup that the file was not found *bios.bin* must additionally with the QEMU start *option-L* is the path to this file will be given. If the file *bios.bin* in the current path point is indicated.

```
Host ~ $ qemu-hda-cdrom ReactOS.img ReactOS.iso-boot c-L.
```

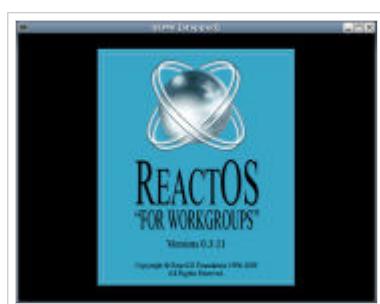
It will start the virtual PC in a window. Installing ReactOS is started by clicking in the window and pressing any key. First, the German keyboard is selected. Thereafter, the virtual disk is partitioned and formatted. The target directory for the installation of the ReactOS files need not be changed. After the



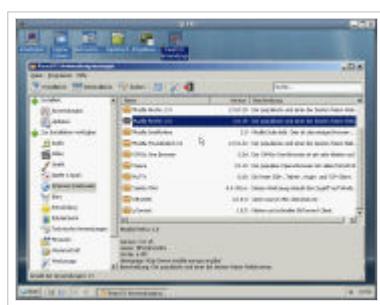
The virtual PC is booted.



bios.bin may need to be made when starting QEMU to the directory containing the file.



ReactOS boot.



ReactOS under QEMU.

boot loader is installed, the default option *Install bootloader on the harddisk (MBR) holds*. Then the virtual machine must be restarted and the installation continues graphically. In order to boot from the hard disk, *c* is the *option-boot* to apply.

```
Host ~ $ qemu-hda ReactOS.img-cdrom-boot c ReactOS.iso
```

To boot without the CD image but with a supported network card, the command applies. This QEMU configured via the integrated DHCP server, the network settings of the host system.

```
Host ~ $ qemu-net user-net ReactOS.img nic, model = PCnet
```

In older QEMU / KVM card is the network versions *ne2k_pci* to use.

```
Host ~ $ qemu-net user-net ReactOS.img nic, model = ne2k_pci
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Quickstart "

This page has been accessed 21,322 times. This page was last updated on 17 March 2010 at 10:31 clock changed. Content is available under GNU Free Documentation License 1.2 .



Translate

Translate from: German

Translate into: English

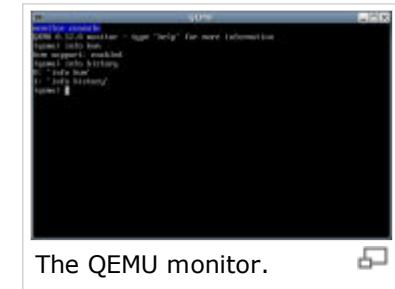
QEMU-/KVM-Instanzen control, QEMU monitor, CD / DVD switch

(Link to this page as [[QEMU-KVM-Buch / The QEMU monitor]])

<<< | # # # | >>> | English

The QEMU monitor

Controlling a virtual machine (instance) whose term is done with keyboard shortcuts and the QEMU monitor.



The QEMU monitor.

Keyboard shortcuts Function

- | | |
|----------------------|---|
| [Ctrl] + [Alt] | Release the mouse and keyboard. |
| [Ctrl] + [Alt] + [1] | Changing the display of the guest operating system. |
| [Ctrl] + [Alt] + [2] | Switch to console 2: QEMU monitor. |
| [Ctrl] + [Alt] + [3] | Switching to console 3: Serial output. |
| [Ctrl] + [Alt] + [4] | Switch to console 4: Parallel output. |
| [Ctrl] + [Alt] + [H] | Provides help in <i>the-Oceanographic</i> . |
| [Ctrl] + [Alt] + [F] | Toggle between full screen and windowed mode. |
| [Ctrl] + [Alt] [U] | Where the original window size restore. |

The QEMU monitor has the following functions:

- Changing or eject removable media (CD / DVD-ROMs, floppy disks).
- The freezing and further run a virtual machine.
- Backing up and restoring various states of the virtual machine.
- Inspecting the state of a virtual machine.
- The migration of a virtual machine to another host.
- Changing the hardware (USB, PCI, ...).
- The injection of emulated hardware failures.

With the key combination [Ctrl] + [Alt] + [2] leads to the QEMU monitor. The precepts of the QEMU monitor on the *help* command lists.

```
(Qemu) help
```

The *info* command gives status information on the current instance. If you give *info* only, parameter is a list of possible output. The QEMU version is *version info* to determine.

```
(Qemu) version info
0.13.0
```

The *kvm* command *info* shows if the KVM hardware virtualization is enabled or not.

```
(Qemu) info kvm
kvm support: enabled
```

The command-line history, the parameter of *history*.

```
(Qemu) info history
```

QEMU-KVM-Buch / Storage Media

For qemu-kvm and libvirt - QEMU + Kernel-based Virtual Machine - Wiki

<<< | # # # | >>> | English

Storage

As a storage media here are hard disks, floppy disks, CD / DVDs, USB memory sticks, called on-board Flash memory, Secure Digital cards, and parallel flashes. For virtual machines instead of real disks usually use image files that are stored in the file system of the host.

- Access to storage media
- Image formats
- Creating Images
- Convert images, compress, encrypt and enlarge
- Images and other virtualizer Emulators
- Physical to virtual
- VM snapshots
- Virtual FAT disk
- Network Block Devices
- Images in the host system include
- Tools for computer forensics

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien "

This page has been accessed 12,969 times. This page was last updated on 26 Amended in February 2010 at 15:18 clock. Content is available under GNU Free Documentation License 1.2 .

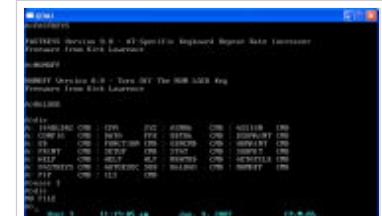
Hard disks floppies DVROM CDROM Memory Cards Flash Memory snapshot.netboot.me

(Link to this page as [[QEMU-KVM-Buch / media / access to storage media]])

<<< | # # # | >>> | English

Contents

- 1 Access to media
 - 1.1 Solid slabs
 - CD / DVD 2.1
 - 1.3 floppy
 - 4.1 Memory Cards and Flash Memory
 - 1.5 boot sequence
 - 1.6 Write Protection
 - 1.7 Definition of drives with the option-drive

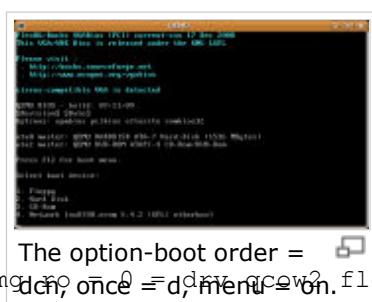


A CP/M-86-Diskette boots in QEMU.

access to storage media

The names and states of the storage media lists in QEMU monitor, the command *info* on *block*. Here is an example. The first column shows the name of the storage medium. Use this name to the media in QEMU monitor is addressed.

```
(Qemu) info block ide0-hd0: type = hd removable = 0 file = os2w4.img
```



The option-boot order =
hd0,hd1,hd2,hd3
option-boot = hd0,hd1,hd2,hd3

Hard Drives

Requests to (virtual) hard drives via the *options-file hda*, *hdb-file,-hdc file*, and *file-hdd-drive show*. This *file* is usually an image file, as this example:

```
Host ~ $ qemu-hda-hdb Platte.img second-Platte.img
```

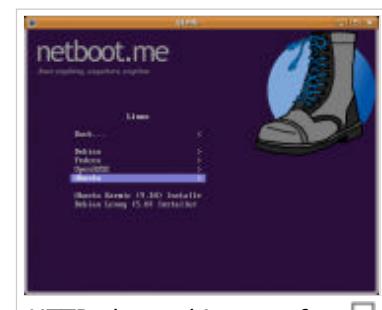
With a single virtual hard disk, the *option-hda* be omitted.

```
Host ~ $ qemu Platte.img
```

It is possible, a hard disk of the host system to provide the guest system. This *file* is the full path to the device specified. In unix-like systems (BSD, Mac OS, Solaris, Linux and others) are the appropriate device files in */dev*. The specific names of the devices are, however, vary on the systems. If the host is a Linux system, these are such as */dev/hda* (first disk on the IDE controller), */dev/sda* (first SCSI disk on controller or SATA controller). Mac OS X addresses the first record */dev/disk0* with. In Microsoft Windows versions are *driven* real *physical* disks with the syntax *\.\.N* involved, where *N* is the number of the drive is (0 is the first hard disk).

When you install Microsoft Windows 2000 as guest operating system is the *option-win2k-hack* needed because the bug hard drive when you install a fully characterizes the. After installation, this option is removed.

aged hard drives that are up to a few hundred megabytes of storage capacity includes the *case*,] the *option-hdachs c, h, s, [, t* interesting. It defines the physical geometry of the first virtual hard drive. It is possible the number of cylinders ($1 \leq c \leq 16\,383$), heads ($1 \leq h \leq 16$) and the sectors ($1 \leq s \leq 63$) set the. Optional is the BIOS translation mode (*t = none, lba* or *auto*) set. The BIOS translation was used to overcome the 524-MB limit on older hard drives and older BIOS versions.



HTTP-charged image of netboot.me.

CD / DVD

Requests to (virtual) CD-/DVDs enable the options and *file-cdrom-drive*, where *file* is an image file or an actual device is.

```
Host ~ $ qemu-hda-cdrom Platte.img cd.iso
```

A CD / DVD device system to the host system to make available the host, is to *file* the full path to the CD / DVD device indicated. If the host is a Linux system, these are such as */dev/cdrom* or */dev/dvd*. The following example will boot a system from a DVD disc.

```
Host ~ $ qemu-cdrom / dev / dvd
```

On Microsoft Windows CD / DVD drives on the host system are integrated by specifying its drive letter. This is certainly familiar to Windows users than the Linux-writing, which is also valid. The following command will boot a CD / DVD from the *D:* drive.

```
Host C: \> qemu-L. Cdrom-d:
```

A CD / DVD to remove or replace it alternates with [Ctrl] + [Alt] + [2] in the QEMU monitor. First, the name of the CD / DVD drive is to be determined.

```
(Qemu) info block  
ide1-cd0: type = cdrom removable = 1 locked = 0 file = os2-warp ro = 0 drv = raw
```

In this example, *cd0* is the CD / DVD drive to *ide1-known*. A CD / DVD release, the command *eject*.

```
(Qemu) eject ide1-cd0
```

A CD / DVD is changing with the *change* command. In this example, a real CD in Linux is changed.

```
(Qemu) change ide1-cd0 / dev / cdrom
```

Was QEMU / KVM support with *curl* compiled, protocol, access is possible through HTTP. In this example image is the instance with the *CD-www.netboot.me* HTTP started from. This can be installed several Linux and BSD distributions.

```
Host ~ $ qemu-cdrom http://static.netboot.me/gpxe/netbootme.iso
```

Floppy

Requests to (virtual) disks allow the *options-fda file* (drive A),*-fdb file* (drive B) *and-drive*. The following example uses a disk image.

```
Host ~ $ qemu-fda Diskette.img
```

Under Linux, disk drives are their real devices (*/dev/fd0*, */dev/fd1*) integrated over. Subsequently, the boot process from the first floppy drive is initiated.

```
Host ~ $ qemu-fda / dev/fd0
```

To remove a floppy disk or replaced as you change the [Ctrl] + [Alt] + [2] in the QEMU monitor. First, the name of the drive is to be determined.

```
(Qemu) info block  
floppy0: type = floppy removable = 1 locked = 0 file = d.dsk ro = 1 drv = raw
```

In this example, drive is the *floppy-floppy0* designated. A floppy disk, the command *eject* freely.

```
(Qemu) eject floppy0
```

Change is a floppy disk with the command *change*. In this example, a real disk in Linux is changed.

```
(Qemu) change floppy0 / dev/fd0
```

Was QEMU / KVM support with *curl* compiled, protocol, access is possible through HTTP. In this example

image is the instance with the floppy www.netboot.me HTTP started from. This can be installed several Linux and BSD distributions. Since HTTP is no write access is possible, a write protection is necessary (see below).

```
Host ~ $ qemu-fda http://static.netboot.me/gpxe/netbootme.dsk \
-Snapshot
```

In the currently used formats, such as introduced by the MS-DOS FAT file system, the master boot record (MBR) for BIOS-based computers of the x86 architecture, the first data block (512 bytes). Is marked by the signature of the MBR *0xAA55*. this signature is present, the BIOS assumes that it is a valid MBR. If the signature is not found, the boot process is interrupted and an error message such as *non-system* or *Non-Bootable Disk* appears. In older disk formats that signature was still unknown. These disks are thus not recognized as bootable, although they may contain an operating system. Nevertheless, to allow a boot, the *option-no-fd-bootchk* apply. They disabled the boot signature test. This makes it possible to boot a CP/M-86-Diskette.

```
Host ~ $ wget http://www.gaby.de/ftp/pub/cpm/sysdisks/cpm86/86raw144.zip
Host ~ $ qemu-fda 144cpm86.img-no-fd-bootchk \
Rtc-base = localtime-m 2
```

Memory Cards and Flash Memory

With the *option-sd* card *file* is a Secure Digital emulated. The file specified *file* serves as a Secure Digital card image.

```
Host ~ $ qemu Platte.img-sd file
```

The *option-mtdblock* *file* memory is an on-board flash emulated. The file specified *file* serves as an on-board flash memory image.

```
Host ~ $ qemu-Platte.img mtdblock file
```

The *option-file pflash* a flash emulates parallel. The file specified *file* serves as a parallel flash image.

```
Host ~ $ qemu-system-arm-Platte.img pflash file
```

Boot sequence

The *option-boot* device from which regulates or image of the boot process is initiated. If more than one after the boot disks are used, the *drives*, the parameter *order* = apply. The following example will first try to boot from the floppy disk. This is not possible, it tries to CD / DVD-ROM to boot. If still unsuccessful, the boot process will start from the hard disk.

```
Host ~ $ qemu-fda Diskette.img-hda-cdrom Platte.img cd.iso \
Boot-order = adc
```

once with the parameter = *drives* is the first time just trying to boot from the media. In the following example, only the boot from the CD / DVD drive is initiated. After rebooting the host system the hard drive as a boot medium is used.

```
Host ~ $ qemu-hda-cdrom Platte.img cd.iso-boot once = d
```

A boot menu to select the boot devices will be activated on the parameter *menu* =. This menu is called in the starting instance using the [F12].

```
Host ~ $ qemu-hda-cdrom Platte.img cd.iso-boot menu = on
```

In older versions of QEMU followed the-boot only one letter (*a*, *c*, *d*, *n*, *o* or *p*). If this option after a given, u is the operating system tries from the first disk to start. *B* with the second floppy drive is specified.

```
Host ~ $ qemu-fda Diskette.img Platte.img-hda-boot a
```

A *c* defines the first hard drive as a boot medium (default). The letter *d* enabled CD / DVD drive. With the letters *n*, *o*, or *p* booting is configured on the network (see *network services*). Here is an example of booting

from the hard disk.

```
Host ~ $ qemu-hda Platte.img-cdrom-boot c cd.iso
```

The QEMU monitor to change a new boot sequence with the command *boot_set*. This overrides the values of *boot*. The values correspond to the parameters *of-boot*, but may vary according to machine type. In the following example, *c-media* set as boot.

```
(Qemu) boot_set c
```

Read-

Storage media can be protected from changes. This causes *the-snapshot*. The name of the option is misleading *snapshot*. With this option, not just a snapshot of the system is generated, but all media is provided with a read-only. Snapshots of the system are created with the other hand, VM snapshots. In the application of *be-snapshot*, the changes actually written to the storage media, temporary files stored in.

Example:

```
Host ~ $ qemu-snapshot Platte.img
```

With [Ctrl] + [Alt] + [s] at any time save the changes to the disk is possible. Likewise, stores the *commit* command in QEMU monitor the changes. The *commit* command *all* causes all devices to be secured. This represents the key combination [Ctrl] + [Alt] + [s].

```
(Qemu) commit all
```

The following command saves only the first disk (*hda* Device).

```
(Qemu) commit hda
```

Definition of drives with the option-drive

From version 0.9.1 QEMU with its *option-drive* options for flexible definition of virtual drives.

```
-Drive [file = file] [, if = type] [, bus = n] [, unit = m] [, media = d] [, index = i] \
[, Cyls = c, heads = h, secs = s [, trans = t]] [snapshot = on | off] \
[, Cache = write through | writeback | unsafe | none] [, format = f] \
[, Serial = s] [, addr = A] [, id = name] [, aio threads = | native] \
[, Readonly = on | off]
```

Possible options are:

```
file = file
```

This option defines which image is used for this drive. If the file name by a comma, it should be double this.

```
if = type
```

This option defines what kind of interface the drive is connected. Options are: *ide*, *scsi*, *sd* (Secure Digital Card), *mtd* (on-board flash memory), *floppy*, *pflash* (parallel Flash), *virtio* (paravirtualized block device).

```
bus n = unit = m
```

These options define by specifying the bus number and unit ID as the drive is connected.

```
media = d
```

This option sets the media type (*disk* or *cdrom*).

```
index = i
```

Defines an interface on which connector is connected the drive. With Connectors, the *index* number of the specified.

cyls = c, heads = h, secs = s [, trans = t]

These options have the same meanings as *in-hdachs*.

snapshot = on | off

The option can be set *on* or *off* and allows the (de) activation of the snapshot function for the specified drive (see *option-snapshot*).

cache = none | write through | writeback | unsafe

The *cache* option can be *none*, *write back*, *write through* (default) or *unsafe* to be set. It controls the use of the cache for the specified drive. With the contact of the parameter *write through* write requests, the host cache for read and used. The confirmation of the write operation is sent to the host system only when the storage subsystem has confirmed the writing process. In the *write back* option is already writing the confirmation sent when the data of the host are stored in the cache. Crashes the host, it can cause loss of data. With *unsafe* contents of the cache never written to the disk. For problems with the host thereby threatening loss of data. At the start *snapshot default* option is *unsafe* as a set. When *none* option cache, the host is not used. QEMU and KVM can cache the data internally. The performance can be certain drivers in combination with the option to *write through* and the image size be less *qcow2*. If more emphasis on speed to put as security, is to apply the *write back* option.

format = f

Specifies the image format (eg *format = raw*).

serial = s

Defines the number for the allocation of the device.

addr = A

Defines the PCI device address of the controller (*virtio*).

boot = on | off

If the option is set *boot = on*, boot the drive is made possible by.

aio threads = | native

Allows you to choose from *Pthread based disk I / O* and *Native Linux AIO*.

readonly = on | off

With *readonly = on* the drive is read-bear.

Instead of the *option-cdrom*, it is possible to use the following options:

Host ~ \$ **qemu-drive file = cd.iso, index = 2, media = cdrom**

The *options-hda-hdb,-hdc and hdd* are replaced as follows:

```
Host ~ $ qemu-drive file = Platte.img, index = 0, media = disk
Host ~ $ qemu-drive file = Platte.img, index = 1, media = disk
Host ~ $ qemu-drive file = Platte.img, index = 2, media = disk
Host ~ $ qemu-drive file = Platte.img, index = 3, media = disk
```

The options include CD-ROM drive as a slave of *ide0* a:

Host ~ \$ **qemu-drive file = cd.iso, if = ide, index = 1, media = cdrom**

If the option *file* does not define drive is created without an inserted medium.

Host ~ \$ **qemu-drive if = ide, index = 1, media = cdrom**

The following options define a SCSI hard disk with unit ID 6 on the bus 0:

```
Host ~ $ qemu-drive file = Platte.img, if = scsi, bus = 0, unit = 6
```

Instead of *the-fda fdb* or can be used the following options:

```
Host ~ $ qemu-drive file = a.img, index = 0, if = floppy
Host ~ $ qemu-drive file = b.img, index = 1, if = floppy
```

The *index* value is incremented automatically.

```
Host ~ $ qemu-drive file = c.img-drive file = d.img
```

This corresponds to the following command line.

```
Host ~ $ qemu-hda-hdb c.img d.img
```

In the QEMU monitor, add the command *drive_add* the running virtual machine a virtual PCI-drive added. In this example, the image is connected *daten.img*.

```
(Qemu) drive_add dummy if = none, id = mydisk, file = daten.img
```

The options are similar to those of *drive_add-drive*. The syntax is:

```
(Qemu) drive_add [[<domain>:] <bus>:] <slot> \
    [File = file] [, if = type] [, bus = n] \
    [, Unit = m] [, media = d] [index = i] \
    [, Cyls = c, heads = h, secs = s [, trans = t]] \
    [Snapshot = on | off] [, cache = on | off]
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Zugriff_auf_Speichermedien "

This page has been accessed 17,999 times. This page was last updated on 17 January 2011 at 16:59 clock changed. Content is available under GNU Free Documentation License 1.2 .

KVM QEMU qemu-img convert, image, image formats sparse qcow2 raw cow qcow cloop vmdk vpc bochs dmg nbd parallels vvfat

(Link to this page as [[QEMU-KVM-Buch / media / image formats]])

<<< | # # # | >>> | English

Contents

- 1 Supported image formats
 - 1.1 raw
 - 2.1 host_device
 - 1.3 qcow2
 - 4.1 qcow
 - 1.5 cow
 - 1.6 vmdk
 - 1.7 vdi
 - 1.8 cloop
 - 09.01 nbd
 - 1:10 parallels
 - 1:11 vvfat
 - 1:12 vpc
 - 1:13 bochs
 - 1:14 dmg

Supported image formats

QEMU and KVM can not only create new virtual storage devices, storage media but also other virtualization programs, such as VMware convert. This tool is described below, the *qemu-img*. QEMU / KVM following image formats are supported.

Raw

The raw format is the default format. It is simple and can easily convert. a new, blank image in raw format to a certain size, for example, produced 10 GB, it occupies in older file systems exactly this size (10 GB). Newer file systems support more effective management of files, containing the uncovered blocks. Such files occupy in these file systems such as just the space they occupy on blocks. A file of unused blocks is **sparse** file. Images in *raw* format files can be created as sparse.

host_device

For block or other devices, the effective management of unused blocks in files do not support, the format *host_device* apply.

qcow2

The format is the most versatile *qcow2* format and replaces the old format from *qcow*. Images in *qcow2* format are dynamic. Their size depends on their level. When *qcow2* format file size is independent of whether the file system for efficient management of sparse files supported. *Qcow2* supports multiple storage of system states of the virtual machine (VM snapshot). Furthermore, encryption (AES) and compression (zlib) is possible. In *qcow2* content is stored in the clusters. Each cluster contains a number of 512 byte sectors. There are options of *qemu-img* (-o) supports the following:

```
backing_file = filename
```

Filename of the base image (see-*img create qemu*).

`backing_fmt = fmt`

Image format of the base image (see-*img create qemu*).

`encryption = on | off`

Encrypted image. The encryption is done with AES (128 bit key), making it very safe. For maximum security, a password must be used with 16 characters.

`cluster_size = aalue`

Changes the default cluster size (512 bytes). There are values between 512 and 2MByte possible. Smaller values optimize the file size. Larger values result in better performance.

`preallocation = off | metadata`

An image with assigned meta-data is initially larger, but has a better performance when zooming.

qcow

qcow is the old QEMU image format. Images in *qcow* format are dynamic. Their sizes depend on their level. In *qcow* the file size does not depend on whether the file system for efficient management of sparse files allowed. Still supported *qcow* encryption and compression. There are options of *qemu-img (-o)* supports the following:

`backing_file`

See *qcow2*.

`encryption`

See *qcow2*.

cow

The old format copy-on-write is available only because of compatibility with older versions of QEMU. It does not work under Microsoft Windows.

vmdk

vmdk is the standard format of VMware Workstation. There are options of *qemu-img (-o)* supports the following:

`backing_fmt`

See *qcow2*.

`compat6`

Forces at the *vmdk* format compatibility level 6 for the destination image file.

vdi

vdi is the standard format of VirtualBox.

loop

The format *Compressed loop* is compressed CD-ROM images used, for example Knoppix-CD/DVD-ROMs.

nbd

nbd), the format of the NBD protocol (Network Block Device.

Parallels

Parallels is the standard format of virtualization solutions from Parallels, Inc.

vvfat

With *vvfat* disks are virtual FAT meant.

vpc

VPC is the standard format for images of Microsoft Virtual PC.

bochs

bochs is the format of the free and AMD64 x86 emulator Bochs.

dmg

dmg is a disk image file format under Mac OS X and is used mostly for distribution of software over the Internet. These images can be uncompressed or compressed. Uncompressed dmg files can be read directly from *qemu img*. Compressed dmg files can be unpacked *dmg2img* the tool with.

<<< | # # # | >>> http://qemu-buch.de

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Image-Formate "

This page has been accessed 12,657 times. This page was last updated on 22 January 2011 at 13:52 clock changed. Content is available under GNU Free Documentation License 1.2 .

Images create qemu-img create, kvm-img dd iso raw qcow qcow2 overlay ovl, sparse image, sparse files, genisoimage

(Link [Create [QEMU-KVM-Buch / Media / Images]] to this page)

<<< | # # # | >>> | English

Contents

- Creating an image
 - 1.1 Creating images with qemu-img
 - 2.1 Creation of raw images and sparse files dd
 - 1.3 Effective Copying Sparse files
 - 4.1 overlay images to create
 - 1.5 CDs, DVDs and floppy image as import
 - 1.6 Creating a CD image from a directory

Creating Images

Virtual machines typically use image files as virtual media. These can be transferred to other computers, making them more versatile than the also possible use real hard drives. After creation, these image files, such as real media, partitioned and formatted. This can be done through programs of guest systems.

Creating images with qemu-img

The QEMU / KVM supplied command line tool *qemu* or *kvm-img-img* is the parameter to *create* image files for the generation of times. According to this parameter, the file name and the virtual image size is specified.

```
Host ~ $ qemu-img create Platte.img 1G
Formatting 'Platte.img', fmt = raw, size = 1048576 kB
```

Here, the virtual size is specified with a GB. In the file system of the host system, the generated file is recorded with a size of a GB. The reason is that without explicit specification of an image format created automatically the image in raw format. The size of the file is determined by the tools of the operating system. On Microsoft Windows does this command you, or the file manager. In these examples Unix-/Linux-Befehle be applied.

```
Host ~ $ ls -lh
-rw-r - r - I 1 I 1,0 G 2007-06-23 11:11 Platte.img
```

For more information on image files with *qemu-img* is obtained and the parameter *info*. As a result of the file name, image format, the virtual size and stored in the file system size is output.

```
Host ~ $ qemu-img info Platte.img
image: Platte.img
file format: raw
virtual size: 1.0G (1073741824 bytes)
disk size: 0
```

Below is an image in qcow2 format can be created. Is defined in the image *qemu-img-format* with the *-f* parameter and specify the desired format.

```
Host ~ $ qemu-img create-f qcow2 plate qcow2.img 1G
Formatting 'plate qcow2.img', fmt = qcow2 size = 1073741824 encryption = off cluster_size = 0
```

With *qemu-img* and the parameter *check* you check images in the format qcow2.

```
Host ~ $ qemu-img-plate check qcow2.img No errors were found on the image.
```

With *qemu-img* and the *info* parameter is allowed to view information about this image.

```
Host ~ $ qemu-img info Platte qcow2.img
image: top-qcow2.img
fileFormat: qcow2
virtual size: 1.0G (1073741824 bytes)
disk size: 140K
cluster_size: 65536
```

In the file system a file size of only 256K KByte appears, although a GB was set as a virtual size.

```
Host ~ $ ls -lh plate qcow2.img
-rw-r - r - 1 256K 2009-11-14 13:32 I-Drive qcow2.img
```

It seems the image file in qcow2 format is much smaller than the image file in raw format. Is that really so? To this end an experiment on a Linux machine. In this example, the /tmp directory as an extra with a size of 1.9 GB partition involved. This shows the command *df*.

```
Host ~ $ df -h | grep "/ tmp"
Filesystem Size Used Avail Use% Mounted on
/ Dev/hda3 1.9 G 6.1 M 1.9 G 1% / tmp
```

The *mount* command determines the file system.

```
Host ~ $ mount | grep "/ tmp"
/ Dev/hda3 on / tmp type ext4 (rw, commit = 0)
```

The underlying file system supports EXT4 effective management of sparse files. For a test is placed in this directory a large image file in raw format.

```
Host ~ $ qemu-img create-f raw / tmp / GrosseImageDatei.img 1000G
Formatting '/ tmp / GrosseImageDatei.img', fmt = raw, size = 1048576000 kB
```

The file with a size of a TB is created without error.

```
Host ~ $ ls -lh / tmp / GrosseImageDatei.img
-rw-r - r - 1 I 1 1000G 2007-06-23 12:53 / tmp / GrosseImageDatei.img
```

Also *qemu-img info* is generous to them.

```
Host ~ $ qemu-img info / tmp / GrosseImageDatei.img
image: / tmp / GrosseImageDatei.img
file format: raw
virtual size: 1.0T (1,073,741,824,000 bytes)
disk size: 0
```

While a file was created with the size of a TB in a partition with 1.9 GB, the level of that partition has not changed.

```
Host ~ $ df -h | grep "/tmp"
Filesystem Size Used Avail Use% Mounted on
/ Dev/hda3 1.9 G 6.2 M 1.9 G 1% / tmp
```

This file will grow in size when it is filled with content. A host system that uses this image has stored any information about the size of the embedded partition of the host system in which this image. If the image is too large for this partition, there is no warning Write error in the host system.

In addition there arises another problem. If you copy this image to a file system that does not support effective management of sparse files, the copy process stops with an error message when the target medium, the image can include in its full size. This can be confusing if not as a backup is successful, although the target medium has a higher capacity. For example, the failed attempt also suggests that image on a USB flash drive or to copy a disk, for these are usually in the *vfat* file system created.

Images of raw and sparse files created with dd

The following examples of creating files with *dd* help to understand how raw images and sparse files are structured. *Dd* is a tool to redirect data streams. This standard tool is available on almost all Unix / Linux systems. *Dd* is also available for Microsoft Windows. This *dd* image files can be created just like the part with *qemu-img* files. a data stream will be diverted from all zeros with a defined length in a file that creates a raw image. The following command creates an image in raw format with a size of one GB. It will be out of the device */dev/zero* zeros got there and the file written in *Platte.img*. The block size of data is set at 1024 KB. *count* in this case indicates how many of these blocks, the parameters are written. A thousand units this translates as a GB.

```
Host ~ $ dd if = / dev / zero of = Platte.img bs = 1024k count = 1000
1000 +0 records in
1000 +0 records out
1048576000 bytes (1.0 GB) copied, 59.4961 seconds, 17.6 MB / s
```

The command *qemu-img* shows that it is really a raw-image concerns.

```
Host ~ $ qemu-img info Platte.img
image: Platte.img
file format: raw
virtual size: 1.0G (1048576000 bytes)
disk size: 1.0G
```

Since this image has been filled with zeros, it is not a sparse file. To generate a sparse file parameter is the required *seek*, the end of the file determines the. Thus, only a certain number of blocks is written in the beginning of the file and record the real end of file. In between, no blocks are written. The actual area occupied by the file space in the file system is correspondingly smaller. The following example will block only a written (*count* = 1) and file the end of 1024 set the blocks.

```
Host ~ $ dd if = / dev / zero of = bs = 1024k count Sparse.img = 1 seek = 1024
1 +0 records in
1 +0 records out
1048576 bytes (1.0 MB) copied, 0.003086 seconds, 340 MB / s
```

The command *qemu-img* shows the virtual and the real size of this file.

```
Host ~ $ qemu-img info Sparse.img
image: Sparse.img
file format: raw
virtual size: 1.0G (1074790400 bytes)
disk size: 1.0M
```

Effective Copying Sparse files

On Unix / Linux supports the command *cp* effective copying sparse files. Default command sparse files from the recognized *cp* and the target files are also created as sparse files. This behavior is with the *-selected sparse = auto*. It is *one-sparse = always* to file a sparse file to generate a target if the source file long enough sequence of zero bytes contains a. We *used-sparse = never* to a sparse file to prevent the production. To test these options, shall we *dd* a sparse file with.

```
Host ~ $ dd if = / dev / zero of = bs = 1024k count Platte.img = 1 seek = 1024
```

With *qemu-img info* you can get information to show that image.

```
Host ~ $ qemu-img info Platte.img image: Platte.img file format: raw virtual size: 1.0G (1074790400 bytes) disk size: 1.0M
```

The actual file size here is a MB. The volume of contrast, is a GB. This file is *cp* and the *-sparse = never* copied.

```
Host ~ $ cp - sparse = never Platte.img plate never.img
```

The copy is with *qemu-img info* investigated.

```
Host ~ $ qemu-img info Platte never.img
image: top-never.img
file format: raw
virtual size: 1.0G (1074790400 bytes)
disk size: 1.0G
```

As expected, the real size is now a GB. In the next example, the original sparse file with the *-sparse = always* copied.

```
Host ~ $ cp - sparse = always Platte.img plate always.img
```

The real size of the file has been reduced since the first MB was all zeros. These zeros are converted to holes.

```
Host ~ $ qemu-img info Platte always.img
image: top-always.img
file format: raw
virtual size: 1.0G (1074790400 bytes)
disk size: 0
```

The *file-disk never.img* holes were in the copy process is converted into the zeros, so the file size of the real one GByte obtained. With the *-sparse = always*, the copy of this file as a sparse file created will be again.

```
Host ~ $ cp - sparse = always disk-drive-never-never.img always.img host ~ $ qemu-img info plate-never-always.img image: Plate-never-always.img file f
```

Images Create overlay

QEMU and KVM support overlay images. These are based on a previously created image file. An overlay image saves the changes to the original image file. The base image is not altered, except in the QEMU monitor, the command is applied *commit*. Contains the base image a complete installation of an operating system, it can serve as a template for additional virtual machines. This save your changes into the corresponding overlay images. When you create the overlay file size must not be defined. The path and name of the base-image is specified with *the-b*.

```
Host ~ $ qemu-img create-f-b Platte.img qcow2 Platte.ovl
```

The command *qemu-img info* shows an overlay image of the virtual size and the path name of the base images on to.

```
Host ~ $ qemu-img info Platte.ovl
image: Platte.ovl
fileFormat: qcow2
virtual size: 650M (681,574,400 bytes)
disk size: 4.0K
```

```
cluster_size: 512
backing file: Platte.img (actual path: Platte.img)
```

The overlay image is embedded as a normal image.

```
Host ~ $ qemu Platte.ovl
```

The image stored in the overlay changes can be written *qemu-img* in the base image with it. The overlay image and the base image may not be in a virtual machine use.

```
Host ~ $ qemu-img-f commit qcows2 Platte.ovl
Image committed.
```

Overlay files make it possible to boot simultaneously from a base image multiple virtual machines.

```
Host ~ $ qemu-img create-f-b Platte.img qcows2 Platte1.ovl
Host ~ $ qemu-img create-f-b Platte.img qcows2 Platte2.ovl
Host ~ $ qemu Platte1.ovl
Host ~ $ qemu Platte2.ovl
```

The following is a script to start when an overlay file from a base image (Variable *BASE_IMG*) port. The name of the overlay image is formed from the name of the base image and a time stamp. Optional when calling the script as parameters a word (without spaces or special characters). This word is integrated by the script in the name of the overlay file.

```
# / Bin / sh
BASE_IMG = "Platte.img"
DATE_TIME = `date '+%Y-%m-%d-%H-%M'`
OVL_NAME = "$BASE_IMG $1 - $DATE_TIME.ovl"
qemu-img create-b "$BASE_IMG" qcows2 f "$OVL_NAME"
qemu-hda "$OVL_NAME"
```

These lines are as a script file named *qemuovl.sh* the directory that contains the image contains, stored in base. Then gives you the script file execute permission.

```
Host ~ $ chmod +x qemuovl.sh
```

When you call this script, a parameter can be specified to indicate this overlay images, such as:

```
Host ~ $ ./ Qemuovl.sh testSP2
```

DVDs and floppy disks as image import CDs

It is possible to generate any modern CD-/DVD-Brenn-Programm image files from CDs and DVDs. Importing an installation media as an image is evaluated from a legal point of view as making a copy of software. It should be therefore subject to the licensing provisions.

On Unix / Linux, it is easy to image files from CDs, DVDs, floppy disks and partitions to create on the command line. This command is the Unix *dd*. Since the names for the floppy, CD and DVD devices in each system may differ, these names are first determined. A look at the file */etc/fstab* or */etc/vfstab* (Solaris) provides information on the system used with the devices.

```
Host ~ $ cat / etc / fstab
/ Dev / hdc / media/cdrom0 udf, iso9660 user, noauto 0 0
/ Dev/fd0 / media/floppy0 auto rw, user, noauto 0 0
```

In this example, *hdc* is the floppy drive on the device */dev/fd0*, and the CD drive on the device */dev* / addressed. When importing the devices may not be integrated by the system. The following command imports a disk:

```
Host ~ $ dd if = / dev/fd0 of = floppy-image.img
```

This command imports an inserted CD.

```
Host ~ $ dd if = / dev / hdc of = cd-image.iso
```

In some versions of Unix, like FreeBSD, the *block size* be specified.

```
Host ~ $ dd if = / dev/acd0 file.iso of = bs = 2048
```

And Mac OS X, there is the *dd* command. An inserted CD / DVD is the system to */Volumes / <name>* included automatically. The name of the CD in the *Finder* displays it. In this example, the *W2P_DE*. To read the CD first the appropriate device is to be identified. In Mac OS X files are the device appears *dev* / under. Interestingly, the devices */dev / disk **, since these disks and CD drives used to be. No CD is inserted, for example, the following entries are related to:

```
Host ~ $ ls / dev / disk *
/ Dev/disk0 / dev/disk0s1 / dev/disk0s2
```

This is the first hard drive *disk0*. This here contains two partitions, *disk0s1* and *disk0s2*. If a CD is to be */dev* automatically inserted a new device.

```
Host ~ $ ls / dev / disk *
/ Dev/disk0 / dev/disk0s1 / dev/disk0s2 / dev/disk1 / dev/disk1s0
```

The CD drive is here */dev/disk1s0* raised with. Involved in the state program the system refuses the *dd* to access the drive. So first, the drive will hang out again.

```
Host ~ $ sudo umount / Volumes/W2P_DE
```

Thereafter, *dd* the contents of the CD and read out the image written to.

```
Host ~ $ dd if = / dev/disk1s0 of = / pdb / data / cd-image.iso
```

At the end you should mount the CD again, otherwise the ejection of the CD for the desktop still located or icon with the *finder* not on the work.

```
Host ~ $ sudo mount_cd9660 / dev/disk1s0 / Volumes/W2P_DE
```

The importing of CDs, DVDs and floppy disks in Microsoft Windows is possible only with additional software. The QEMU Manager for Windows allows you to import from floppy disks, CDs and DVDs as image files with a wizard. This will be started in Qemu Manager for Windows on an icon. It will ask if you want to import a CD / DVD or floppy disk. In the next window will ask for the source drive and it must be a file name for the image file can be specified.

The program *dd* there is not only on Unix and Linux, but it is also available for Microsoft Windows. *Dd* can be used under *Cygwin*. But you can also install a separate tool *dd*. It is from the URL <http://www.chrysocome.net/downloads/dd-0.5.zip> downloaded, unpacked and called directly. The following description refers to the latter variant. At first, the internal name of the CD drive is detected.

```
Host C: \ tools \ bin> dd.exe - list
\ \. \ Volume {c7ef4360-d6e9-11d8-ab2a-005056c00008} \
link to \ ? \ Device \ CdRom0
CD-ROM
Mounted on w: \
```

The CD is here in drive *W*: and the internal name is \ \ CdRom0? \ Device \ . This device can be read in a file.

```
Host C: \> dd.exe if = \ \ Device \ CdRom0 of = c \ cd-image.iso
```

Creating a CD image from a directory

On Linux, from a directory with contents a CD image can be generated. The necessary tool *genisoimage* (former *mksisofs*) allows the cdrkit suite (<http://www.cdrkit.org> or be installed separately). On Debian or Ubuntu installation is done with a command line.

```
Host ~ $ sudo apt-get install genisoimage
```

The creation of the CD image made with the following command.

```
Host ~ $ genisoimage-R-J-o CD-image.iso Directory
```

This CD image can be used normally.

```
Host ~ $ qemu-hda-cdrom CD-Platte.img image.iso
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Images_anlegen "

This page has been accessed 16,893 times. This page was last updated on 22 January 2011 at 13:59 clock changed. Content is available under GNU Free Documentation License 1.2 .

qemu-img convert, encrypted, gparted live cd, converting images compress, encrypt and enlarge

(Link to this page as [[QEMU-KVM-Buch / media / Convert Image Files]])

<<< | # # # | >>> | English

Images compress, convert, encrypt and enlarge

The tool *qemu-img* can convert images into different formats. The purpose of the parameter *convert*. The format of the target system is specified with *the-o*. The source format is used by *qemu-img* usually automatically detected, it can also be specified with the parameter *f*. Here is example of a virtual disk in raw format in the format *qcow2* converted to.

```
Host ~ $ qemu-img convert-f raw-O qcow2 Platte.img plate dyn.img
```

The image formats *qcow* *qcow2* and can be personalized with *qemu-img* and the *compress-c* parameter. If, after compressing sectors rewritten these are not compressed.

```
Host ~ $ qemu-img convert-c-O qcow2 Platte.img komprimiert.img
```

The image formats *qcow* *qcow2* and can be personalized with *qemu-img* and the *parameter-e* encrypt with a password. The encryption is done with AES (128 bit key), making it very safe. For maximum security, a password must be used with 16 characters.

```
Host ~ $ qemu-img convert-e-O qcow2 Platte.img encrypted.img
Disk image 'encrypted.img' is encrypted.
password: *****
```

This image is included as usual. The authority is waiting in the stop state.

```
Host ~ $ qemu encrypted.img
```

In the QEMU monitor, the command *c* the stop state of the instance is terminated and it is the password.

```
(Qemu) c
ide0-hd0 (encrypted.img) is encrypted.
Password: *****
```

The QEMU monitor to change the password for the device with the command *block_passwd*. Previously, the name of the device is determined. In this example, it is the device *ide0-hd0*.

```
(Qemu) info block
ide0-hd0: type = hd removable = 0 backing_file encrypted.img = ro = 0 = drv qcow2 encrypted = 1
(Qemu) block_passwd ide0-hd0 ****
```

If an image file is too small, enlarge it can be. First, the host system to be shut down. Since only images in raw format can be enlarged format is the first to identify and, if necessary, convert the image. *Qemu-img info* determines the format.

```
Host ~ $ qemu-img info Platte.img
FileFormat: qcow2
```

In this example, the image format *Platte.img* *qcow2* in front. It must be converted.

```
Host ~ $ qemu-img convert-O raw-disk Platte.img raw.img
```

12:13 From QEMU is the *command-img* *qemu resize* support. With *resize* format, the capacity of a virtual disk in raw changed. In the following example, the capacity of the image is enlarged by 2 GB. In the host system, this new memory is to partition and format (see below).

```
Host ~ $ qemu-img-resize disk raw.img +2 G
```

In older versions of QEMU the magnification of an image is done by attaching a second image in raw format. If the image can be enlarged for example, 500 MB, another image is to be applied with a size of 500 MB.

```
Host ~ $ qemu-img create-f raw temp.img 500M
```

In Linux, files are merged with several *cat*.

```
Host ~ $ cat plate raw.img temp.img> disk-size-raw.img
```

On Microsoft Windows *b* files in the DOS command prompt using the *copy* command and the */ merged*.

```
Host C: \> copy / b Platte raw.img + temp.img plate-size-raw.img
```

qemu-img info shows changes in the size.

```
Host ~ $ qemu-img info-plate-size image raw.img: plate-size-raw.img file format: raw virtual size: 1.1g (1,205,862,400 bytes) disk size: 1.1g
```

The additional area on the virtual hard disk is partitioned nor formatted. An increase in the first partition to these areas enables the Gnome Partition Editor *GParted* short (<http://gparted.sourceforge.net>). You start QEMU or KVM with the enlarged image and boot from the GParted live CD.

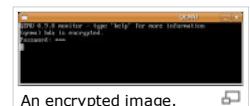
```
Host ~ $ qemu-hda disk-size-raw.img \
-Cdrom gparted-livecd-0.3.4-8.iso std-vga-boot d
```

After startup, the GNOME Partition Editor, the partitions and free areas on the disks. To enlarge a partition, choose it and click on the icon */ size change and move*. A window will appear. There one moves to increase the partition of the arrow on the right or sets the size you want with the numbers. Then you click on the button */ size change and move*. Then, you press *Apply* icon to link to the hard disk to write the changes. To finish the GParted live CD with a click on the icon *Exit*.

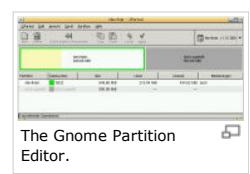
<<< | # # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Konvertieren_von_Image-Dateien"

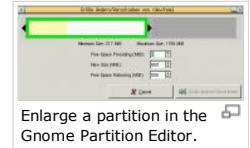
This page has been accessed 13,679 times. This page was last updated on 22 January 2011 at 14:18 clock changed. Content is available under GNU Free Documentation License 1.2 .



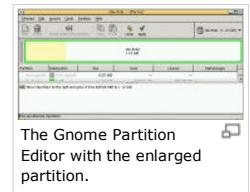
An encrypted image.



The Gnome Partition Editor.



Enlarge a partition in the Gnome Partition Editor.

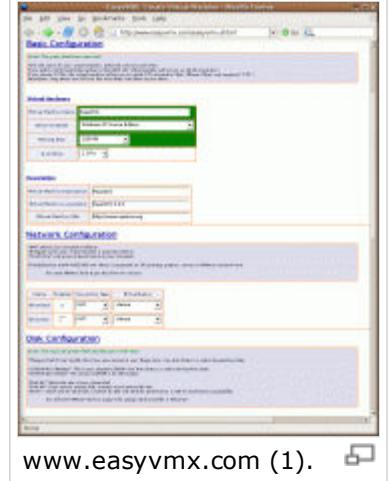


The Gnome Partition Editor with the enlarged partition.

convert qemu, qemu virtualbox, vmdk, bochs, photos and other virtualizer emulators, physical-to-Virtual, V2V migration

(Link to this page as [[QEMU-KVM-Buch / media / disk images of other virtualization software]])

<<< | # # # | >>> | English



Contents	
1 Images and other virtualizer Emulators	
1.1 QEMU and KVM	
VirtualBox 2.1	
VMware Player 1.3	
VMware Workstation 4.1	
1.5 Parallels Desktop / Workstation	
1.6 Microsoft Virtual PC 2007	
Xen 1.7	
Bochs 1.8	
1.9 FAUmachine	
1:10 gxemul	

Images and other virtualizer Emulators

QEMU and KVM virtual machines can use images of many other virtualizer and emulators. When booting from external image is important to note that other virtualization solutions emulate other hardware. You must also install the appropriate driver in the host system. The conversion of virtual machines is also called V2V migration.

QEMU and KVM

Since the Kernel-based Virtual Machine one QEMU emulates the hardware, the images are mutually compatible. Using QEMU, you can move virtual machines for the Kernel-based Virtual Machine. This is thus also possible on the hardware, the Kernel-based Virtual Machine is not running. It can be started with the Kernel-based Virtual Machine but only guest systems with x86 architecture. Further on is the version of the Kernel-based Virtual Machine QEMU to respect, since different versions differ QEMU emulated hardware components can.

```
Host ~ $ qemu-hda Platte.img
Host ~ $ kvm-hda Platte.img
```

VirtualBox

http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_VirtualBox

VirtualBox by Oracle operates on the principle of Native Virtualization. VirtualBox source code used by QEMU. Converting images is possible over the raw format. To this end, a tool used by VirtualBox. Here is an example for a Linux convert a QEMU-/KVM-Images in the VirtualBox format.

```
Host ~ $ qemu-img convert-O raw-disk Platte.img raw.img
Host ~ $ VBoxManage internalcommands converttoraw \
          Disk-drive-hd.vdi raw.img
```

From version 2.1 supports the VirtualBox disk image formats VMDK (VMware) and VHD (Microsoft), including snapshots. 12:12 From QEMU can use the tool *qemu-img* to convert the format *vdi*.

```
Host ~ $ qemu-img convert-O qcows2 Platte.vdi Platte.img
Host ~ $ qemu-img convert-O vdi Platte.img Platte.vdi
```

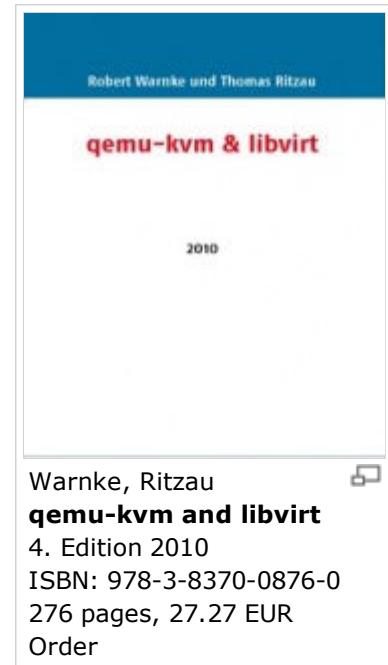
QEMU, the *Virtual Disk Images* (VDI) VirtualBox use directly.

```
Host ~ $ qemu Platte.vdi
```

VMware Player

http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_VMware-Player

The tool *qemu-img* supports *vmdk* format from VMware.



```
Host ~ $ qemu-img convert-O qcow2 Platte.vmdk Platte.img
Host ~ $ qemu-img convert-O vmdk Platte.img Platte.vmdk
```

QEMU can use the images directly from VMware.

```
Host ~ $ qemu Platte.vmdk
```

The free VMware Player is version 3 allows virtual machines to generate. Alternatively, you can move virtual machines with QEMU. The first is for the virtual machine directory to create and change to that directory. In the virtual machine to run ReactOS.

```
Host $ mkdir ~ ReactOS VMPlayer
Host $ cd ~ ReactOS VMPlayer
```

With *qemu-img* ReactOS is an existing QEMU-/KVM-Image of converts.

```
Host ~ $ qemu-img convert-O vmdk ReactOS.img ReactOS.vmdk
```

The virtual machine for VMware Player in addition to the virtual disk configuration file (. *Vmx*). There are free Internet tools that help to generate this file. For this example, EasyVMX (<http://www.easyvmx.com/easyvmx.shtml> serve). You give us the information necessary to configure the virtual machine in the Web form, which will output the contents of the configuration file. First you need to decide the name of the virtual machine and selects the type of the host system. Then we define the size of memory and the number of virtual CPUs. In Section *Description* Comments on the virtual machine can be entered. Under *Network Configuration* *Ethernet0* is the virtual network card is already activated. When selecting *NAT* subnet is a separate formed the *Network Address Translation* to the network of the host communicates. The device *vlance* is supported by most operating systems. Behind this device network card conceals a *pcnet32* based on the *AMD Ethernet controller*. In the *Disk Configuration* section, you define the virtual floppy, CD / DVD drives and hard drives. The floppy disk is in this example, to disable and configure *AutoDetect* to. The first CD-ROM drive to activate and configure *AutoDeect*. The size of the image file for this example is set to 500 MB. All options under *Sound and I / O ports configuration* is disabled in this example.

Virtual Machine Name: **ReactOS**

Select GuestOS: **Windows XP Home Edition**

Memory Size: **128 MB**

Of CPUs: **1 CPU**

Virtual Machine Description: **ReactOS**

Virtual Machine Long Name: **ReactOS**

Virtual Machine URL: **www.reactos.org**

Device Enabled Connection Type virtual device

Ethernet0: **x** **NAT** **vlance**

Device Enabled Floppy Device

Floppy Disk Drive: **Auto Detect**

Device Enabled File Name Device Type

CDROM # 1 **x** **Auto Detect**

Device Enabled Disk Size SCSI? Device Type Disk Mode

Disk # 1: **x** **500MB** **Disk Image (. Vmdk) Persistent**

Then you click on the button *Create Virtual Machine*. The output is copied into a text editor and save as a *vmx* configuration file in the directory created from them. In this example, the file called *ReactOS.vmx*. It is VMware player with this *vmx* file is run the.

VMware Workstation

http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_VMware-Workstation

VMware Workstation is a paid product and works on the principle of Native Virtualization. QEMU and KVM virtual machines can start with the image files from VMware Workstation to Version 4 directly.

```
Host ~ $ qemu vmware-hd.vmdk
Host ~ $ kvm vmware-hd.vmdk
```

VMware knows two basic formats for virtual hard drives: *flat* and *sparse*. An image from a file type is *flat*, the total size of the disk includes. A 10-gigabyte hard drive is here represented by another 10 GB large file. Is the image contrast from the *sparse* type, is the actual disk space used in the file shown only. Large virtual hard disks on VMware products, often multiple image files (. Vmdk), each with a maximum size of two gigabytes divided on. In addition, there is another file (. Vmdk) is created which, however, contains meta-data. The best way to understand the structure of a virtual disk for the example of a VMware virtual machine with an 8-gigabyte hard drive.

Case 1: The entire hard disk to an image file is located, for example *win.vmdk*. It contains all the necessary data and can easily convert to other virtualization solutions. However, was a snapshot in VMware made, this file contains only the state of the file system until the time of the snapshot. All changes made to another file (*win-000001.vmdk*) stored in. Possibly other snapshots are counted (*000002.vmdk* *win*, *win-000003.vmdk* ...). Before converting the data from the base file and snapshots are back together.

Case 2: The image file is divided into slices. In this example, next to the file in addition to files *win.vmdk* *win-win-s005.vmdk* *s001.vmdk* to present hard drive. *Win.vmdk* contains meta-data to build the virtual:

```
# Disk DescriptorFile
version = 1
CID = 793c1936
parentCID = ffffffff
create type = "twoGbMaxExtentSparse"
# Extent description
RW 4192256 SPARSE "win-s001.vmdk"
RW 4192256 SPARSE "win-s002.vmdk"
RW 4192256 SPARSE "win-s003.vmdk"
RW 4192256 SPARSE "win-s004.vmdk"
RW 8192 SPARSE "win-s005.vmdk"
# The Disk Data Base
# DDB
ddb.virtualHWVersion = "4"
ddb.geometry.cylinders = "1024"
ddb.geometry.heads = "255"
ddb.geometry.sectors = "63"
ddb.adapterType = "BusLogic"
```

Also in this case, snapshots are possible. Then the files are also *000001.vmdk* *win* and *win-000001-s001.vmdk* to *win-000001-s005.vmdk* before. Again, in *win-only* meta-data contains *000001.vmdk*. To this for QEMU or KVM to produce usable images *vdiskmanager* used the *vmware-tool* that came with the freely available VMware Server (<http://www.vmware.com/products/server/>). It is called both Microsoft Windows and Linux on the command line.

```
Host ~ # vmware-vdiskmanager-r-t 0 000001.vmdk win-win-export.vmdk
```

This command converts the file *win-win* in the export file *000001.vmdk* *export.vmdk*. In this case, a complete file, starting from the first snapshot of type *sparse* (-t 0) is generated. This export file can be used by QEMU or KVM. Conversion of the VMware configuration files (. Vmx) in the configuration files for *libvirt* is the tool *vmware2libvirt* (see '*libvirt*).

Parallels Desktop / Workstation

Website: <http://www.parallels.com>

Parallels Desktop is a commercial virtualization software that is available in Mac OS X. On Microsoft Windows and Linux, the software is offered as Parallels Workstation. Parallels uses a proprietary,

undisclosed format for their virtual disks. This standard format, the image grows with increasing content of the file system until the prescribed maximum value. There is also a raw format. To use images of Parallels under QEMU or KVM, they must be present in raw format. The images always have the extension. *Hdd*. It can be the guy on Unix / Linux command *file* to determine.

```
Host ~ $ file *. hdd
w2k3.hdd: data
w2k3_plain.hdd: x86 boot sector
```

the raw file, in this example, the file *w2k3_plain.hdd* in, the tool detects a boot sector *file*. These raw file can be converted with *qemu img*. The file contains nothing recognizable *w2k3.hdd* and is therefore qualified as *data*. This file must be converted first into a raw file. This is done with the *Parallels Image Tools*.

```
Host ~ $ qemu-img convert-O qcow2 w2k3_plain.hdd w2k3.qcow2
```

The return of QEMU / KVM to Parallels also works. This image must be converted into a raw file. This is either directly or previously included by Parallels *Parallels Image Tools* to a file format changed with variable size.

```
Host ~ $ qemu-img convert-O raw w2k3_plain.hdd w2k3.qcow2
```

Microsoft Virtual PC 2007

http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_Microsoft_Virtual_PC

Microsoft Virtual PC is offered as a virtualization software for Microsoft Windows and as a x86 emulator for Mac OS X. With Virtual PC, a complete PC is virtualized or emulated. QEMU and KVM can run image files from Virtual PC 2007 directly.

```
Host ~ $ qemu-VirtualPC Festplatte.vhdk
```

Xen

http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_Xen

Xen is a type 1 hypervisor that runs directly on x86 hardware. Xen uses source code from QEMU. Xen images in raw format can be integrated in a virtual machine under QEMU or KVM. It should be noted that these images contain only one partition. In the following example, an image of a Xen host system is included as a second hard disk.

```
Host ~ $ qemu-hda hdb Ubuntu.img-xen-gast.disk
```

QEMU or KVM images can be utilized for Xen. First you have to convert to the image in the raw format.

```
Host ~ $ qemu-img convert-O raw hd.img hd.raw
```

Because the virtual disks of QEMU and KVM usually consist of several partitions, the partition will be extracted. These partitions with *fdisk* to determine.

```
Host ~ # fdisk -lu hd.raw
Units = sectors of 1 * 512 = 512 bytes
Device Boot Start End Blocks Id System
hd1 * 63 32 129 16 033 + 83 Linux
32 130 305 234 136 552 HD2 + 82 Linux swap
hd3 305 235 10,474,379 5,084,572 + 83 Linux
```

To extract a partition, one must calculate the number of blocks, the formula is: end - start + 1 In this example, the third partition to be read. It is a block number is 10474379 to 305,235 + 1 = 10,169,145th With these values and the *dd* command partition is the third extracted. The calculated value is the parameter *count* and the start value is the *skip* parameter.

```
Host ~ # dd if = of = hd.raw xen.img bs = 512 skip = 305 235 count = 10,169,145
```

With xenner (<http://kraxel.fedorapeople.org/xenner/>) it is possible Xen-guest systems under the Kernel-based Virtual Machine to run.

Bochs

http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_Bochs

Bochs is an x86 emulator (Open Source), which on many operating systems, and among other processors, running. The QEMU project uses parts of the source code of the Bochs project, such as the PC BIOS. One image for Bochs *bximage* program is created with the. It generates image files in the formats *flat*, *sparse* or *growing*. QEMU and KVM can read these formats and embed directly. Here is an example.

```
Host ~ $ bximage
Do you want to create a floppy disk image or a hard disk image?
Please type hd or fd. [Hd] [Return]
What kind of image should I create?
Please type flat, sparse or growing. [Flat] sparse
Enter the hard disk size in megabytes [10] [Return]
What should I name the image? [C.img] [Return]
```

The program recognizes *qemu-img* that format.

```
Host ~ $ qemu-img info c.img
image: c.img
file format: raw
virtual size: 32K (32768 bytes)
disk size: 32K
```

On the website of the point Bochs *disk images* are images with pre-installed operating systems offered under. These can be used in QEMU and KVM.

FAUmachine

http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_FAUmachine

FAUmachine is both a hardware emulator and a virtualization solution. FAUmachine is open source and runs as a normal user process under Linux (x86). Ports on OpenBSD and Windows are currently being developed. FAUmachine uses parts of the source code of QEMU. The advantages of FAUmachine are:

- It can be emulated hardware failure.
- For automated experiments, tests and installations is an experiment controller.

The start script *faum* can only manage a virtual machine itself. The configuration of the virtual machine, the file *~/.Faumrc* stored in. FAUmachine can only read images in raw format.

```
Host ~ $ source ~/.Faumrc
Host ~ $ qemu-img info $ IMAGE_PATH/node.def/ide_gen_disk-11.media
image: / VMS/faumachine/node.def/ide_gen_disk-11.media
file format: raw
virtual size: 2.0G (2147483648 bytes)
disk size: 2.0G
```

This image can be in the QEMU or KVM used directly.

```
Host ~ $ qemu $ IMAGE_PATH/node.def/ide_gen_disk-11.media
```

It is also possible with *qemu-img* for FAUmachine to create images or convert it.

```
Host ~ $ qemu-img convert -O raw Platte.img \
$ IMAGE_PATH/node.def/ide_gen_disk-11.media
```

gxemul

http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulatoren/_GXemul

Gxemul emulates the following computer architectures: *ARM*, *MIPS*, *PowerPC*, and *SuperH*. Gxemul is open source and can be installed on most Unix-like operating systems. Gxemul used only images in raw format. Other image formats must be converted *img* gxemul with *qemu-by*. With KVM gxemul of images can not boot process must be started as KVM only supports the x86 architecture. QEMU can embed images of gxemul While booting will fail, but by differences in the emulated hardware in most cases. Gxemul QEMU MIPS has a special mode, can be used to run virtual machines emulated by QEMU MIPS architecture. By following the procedure in the boot gxemul *guest systems*, *MIPS processor architecture* described Debian.

```
Host-e ~ $ gxemul qemu_mips-M 128-d DebianEtch_on_MIPS.img \
-O 'console = ttyS0 rd_start = 0x80800000 = 10000000 rd_size init = / bin / sh' \
0x80800000: initrd.gz vmlinuz-2.6.18-4-qemu
```

<<< | # # # | >>> <http://www.qemu-buch.de>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware "

This page has been accessed 26,582 times. This page was last updated on 27 September 2010 at 18:02 clock changed. Content is available under GNU Free Documentation License 1.2 .

Physical-to-Virtual, P2V Migration, p2v converter, p2v assistant

(Link to this page as [[QEMU-KVM-Buch / media / physical-to-Virtual]])

<<< | # # # | >>> | English

Physical to virtual

Old computers not only consume space and power, they must be serviced more often, and the probability of failure increases. A move to a new hardware can be problematic if the old operating system on modern hardware is not running. A clean installation on a virtual machine can be time consuming or even impossible when the media and documentation for installation and configuration is incomplete. Then I transfer the contents of the hard disk of the old computer's operating system, applications and transfer data to an image. This method is also applied to the evidence and investigation after a system break-in. The transfer of an operating system of a real machine on a virtualized machine is called physical to virtual (P2V). There are solutions to commercial. However, these solutions are only available for some operating systems. Here is the installed operating system independent open source solution is described.

The necessary steps are: The old computer is started with a Linux live CD. With the tools *dd*, *netcat* and *gzip* is a dump of the old hard drive and creates compressed over the network to another computer with enough hard drive space transfer. On the target computer, the transmitted data stream is decompressed and saved as an image. QEMU or KVM with the operating system can be started directly on this image. But you can with *qemu-img* the image for another virtualization program convert.

After this brief review now for the precise description of the steps. You start the old computer with the Linux LIVE. This is necessary to ensure that no write operations take place on the hard disk. When choosing a Linux-Live-CD/-DVD is to make sure that the tools *dd*, *gzip* and *netcat* available. In addition, the network card and hard drives are recognized. This will be done, for example the Knoppix live DVD or *grml* (<http://www.grml.org>) in question. The latter can run on older hardware. Is not even a CD-ROM drive available, disk is to use a live store (<http://mulinux.dotsrc.org>). After the system is booted from Live-CD/-DVD, it is important to *fdisk* the device names of hard disks to be determined.

```
~ # Fdisk-1
```

As */dev/hda* is the first IDE drive specifically and */dev/hdb* marks the second IDE drive. SCSI disks on Linux with */dev/sda*, */dev/sdb*, and continue to appear Sun If, for example on the hard disk */dev/hda* to virtualize the system, you have to create an image of it. This requires the program *dd*. The precise statement is in this example:

```
~ # Dd if = / dev / hda of = hda.raw
```

The parameter defines *if* the file to read (in this case the hard disk device), the parameter *of* the write file (here the image). Since this image is probably no place on the old computer has to copy it right over the network to another computer, which in this example has the IP address 192.168.1.123. If no DHCP server automatically assigns IP addresses, the old computers still get an IP address. This is done with the *ifconfig* command.

```
~ # Ifconfig eth0 192.168.1.234 netmask 255.255.255.0 up
```

The computer, which should save the image, this can be received. The old computer, however, must send this file. The data transfer enables the program *netcat* on both sides. *Gzip-compressed* at the transmission, where the parameter *c* to standard output *writes*. Decompressed with *the-d*. *Gzip* checks the correct transfer and check-sums. The computer to store the image is switched to receive.

```
Host ~ # netcat-w30-vvn1 3333 | gzip-dc> hda.raw
```

On the old computer with the live CD will be the transfer of the hard drive.

```
~ # dd if = / dev / hda | gzip-c | netcat-w 30 192.168.1.123 vvn 3333
```

dd / reading this, the hard disk *dev / hda* and sent to standard output through *gzip* to compress. The compressed data stream from the IP address 192.168.1.123 *netcat* to port 3333 sent to. the receiving computer takes the data stream to *netcat* on port 3333, contrary to. *gzip* decompresses this data stream. The output is redirected to the file *hda.raw*. If the transmission to end, this image directly in QEMU or KVM to be used. It is recommended that before a backup of this file to create.

```
Host ~ $ qemu-snapshot hda.raw
```

Perhaps the system will recognize the start of other hardware. That is, you may need to install the appropriate drivers.

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Physical-to-Virtual "

This page has been accessed 17,413 times. This page was last updated on 4 January 2011 at 11:58 clock changed. Content is available under GNU Free Documentation License 1.2 .

VM snapshots, snapshot savevm drive backup, system snapshot, qcow2

(Link to this page as [[QEMU-KVM-Buch / media / VM snapshot]])

<<< | # # # | >>> | English

VM snapshots

VM snapshots to save the states of the entire virtual machine (CPU, memory, devices and recordable discs). In this way, changes can be made to the system back. Older versions of QEMU save VM snapshot in separate files. From the QEMU version 0.9.0 is the storage of the VM virtual snapshots directly on the hard drives. The prerequisite for this is that at least one of the included disk images in the format *qcow2* present.

```
Host ~ $ qemu-img info Platte.img
image: Platte.img
FileFormat: qcow
```

In this example, the image before the *qcow* format. Therefore, it must be converted into the *qcow2* format.

```
Host ~ $ qemu-img convert-O qcow2 Platte.img plate.qcow2
```

The information about the VM snapshots are written to the first virtual hard drive. The VM snapshots themselves are stored on each image, are present in *qcow2* format. The *option-snapshot* but can VM snapshots are generated, but these go to the end of the virtual machine lost after. *The-snapshot* option has nothing herein VM snapshots to do with.

For generating a VM snapshot is used in QEMU monitor the command *savevm*. A VM snapshot should be provided with a name (tag) to identify these later. In addition, each VM snapshot automatically a progressive number (ID). Access to VM snapshot is possible through this ID, as well as his name. There is already a VM snapshot with the same name or with the same number, it will be overwritten.

```
(Qemu) savevm vor_PatchDay
```

After creating the VM snapshots, changes can be made to the system. After that, another VM snapshots are created.

```
(Qemu) savevm nach_PatchDay
```

If it is unwanted changes have come to this command *loadvm* be reversed with the. It restores the state of the specified VM snapshots.

```
(Qemu) loadvm vor_PatchDay
```

No longer needed VM snapshots should be deleted because they take up space.

```
(Qemu) delvm vor_PatchDay
```

Information on existing VM snapshot shows the QEMU monitor, the command *info snapshots*.

```
(Qemu) info snapshots
Snapshot devices: hda
Snapshot list (from hda):
ID TAG VM SIZE DATE VM CLOCK
1 vor_PatchDay 41M 2006-08-06 12:38:02 00:00:14.954
2 nach_PatchDay 40M 2006-08-06 12:43:29 00:00:18.633
```

It is possible for a virtual machine with the state of a VM snapshot start.

```
Host ~ $ qemu-hda-Platte.img loadvm nach_PatchDay
```

The tool *qemu-img* can VM snapshots for inactive virtual machines to manage. The following command stores a VM snapshot with the name *vor_update* qcow2 on the image.

```
Host ~ $ qemu-img disk-snapshot-c vor_update qcow2.img
```

With the command *qemu-img info* you can find information about the image stored in the VM's snapshots.

```
Host ~ $ qemu-img info Platte qcow2.img
image: top-qcow2.img
FileFormat: qcow2
virtual size: 230M (241,172,480 bytes)
disk size: 98m
cluster_size: 4096
Snapshot list:
ID TAG VM SIZE DATE VM CLOCK
Vor_update 1 0 2009-05-19 18:58:33 00:00:00.000
```

The state of the specified VM snapshot provides you with *qemu-img-a snapshot* restore. With the command *qemu-snapshot-d img* do you delete a VM snapshot.

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_VM-Snapshots "

This page has been accessed 11,625 times. This page was last updated on 27 September 2010 at 06:06 clock changed. Content is available under GNU Free Documentation License 1.2 .

Virtual FAT disk

(Link to this page as [[QEMU-KVM-Buch / Media / Virtual FAT hard drives]])

<<< | # # # | >>> | English

Virtual FAT disk

QEMU and KVM can provide a list of the host computer directly as a virtual hard drive in FAT format to the guest. This option allows the *fat*. To this end, create a directory and issue the corresponding access rights (eg on Unix).

```
Host ~ $ sudo mkdir / exchange  
Host ~ $ sudo chown-R `whoami` / exchange
```

Now you start QEMU or KVM use this directory as a second hard disk (*hdb-*). This is the absolute path is specified. Simultaneous requests from the host and the guest system to this directory should be avoided. It may in the virtual disk Fat-only file names are used, which also supports the guest. *The-loadvm* should not be a virtual FAT disk to be used together.

```
Host ~ $ qemu-hda-hdb fat Platte.img: / exchange
```

The virtual FAT disk is write protected. Write access allows the *parameters: rw:*. This parameter is used with *snapshot-together*.

```
Host ~ $ qemu-hda-hdb Platte.img fat: rw: / exchange
```

Virtual FAT *floppy* diskette can be emulated with the parameters.

```
Host ~ $ qemu-hda Platte.img-fda fat: floppy: / exchange
```

Again, the *parameters: rw:* to be applied.

```
Host ~ $ qemu-hda Platte.img-fda fat: floppy: rw: / exchange
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Virtuelle_FAT-Festplatten "

This page has been accessed 7498 times. This page was last updated on 27 Amended in February 2010 at 19:49 clock. Content is available under GNU Free Documentation License 1.2 .

Network Block Devices **qemu-kvm-nbd** **nbd** **nbd-client** **nbd-server**

(Link to this page as [[QEMU-KVM-Buch / Storage / Network Block Devices]])

<<< | # # # | >>> | English

Network Block Devices

Network Block Devices (<http://nbd.sourceforge.net>) are virtual storage devices, the NBD server over TCP / IP to be provided by one. As a Network Block Device (NBD) can be used hard drives, hard disk partition, CD / DVDs, floppy disks or image files on the NBD server. Other computers can connect over a TCP / IP connection to the NBD server and use the Network block devices such as its own local disks. The access speed is limited by the transmission rate in the network, not high. From kernel 2.6.26, Linux supports the Network block devices. The *NBD* kernel module sets to the devices / *dev/nb* * to. On Linux, *qemu-kvm* package includes the specific NBD server *qemu* or *kvm-nbd-nbd*. With these tools, network block devices are provided. With the *-k* socket is a Unix-used. The *-t* option causes the program when the connection is terminated not abort. Here is an example of the call *qemu-nbd*. For *kvm-nbd* options are the same applied.

```
Host ~ $ qemu-nbd-t-k / var / lock / qemu-nbd Platte.img
```

On a second console of the host-guest system, a system of this network block device is started.

```
Host ~ $ qemu-hda nbd: unix: / var / lock / qemu-nbd
```

If a network block device over TCP / IP export, the port is specified. If the default port used 1024, this may be omitted. In the following example, port is the image file *Platte.img* via TCP / IP over the *1025* network block device as a courtesy.

```
Host ~ $ qemu-nbd-t-p 1025 Platte.img
```

On a second console of the host system can boot an instance of that image.

```
Host ~ $ qemu-hda nbd: localhost: 1025
```

Should be accessed from another computer on that network block device, the name or IP address of the NBD server is specified.

```
Host2 ~ $ qemu-hda nbd: my_nbd_server: 1025
```

The option *-shared = num* allows multiple guest systems can use a network block device access to. The maximum number defines the value of *num* (Default = 1).

```
Host1 ~ $ qemu-nbd-t-p 1025 - share = 2 Platte.img
```

On another computer can boot up an instance with the Network Block Device.

```
Host2 ~ $ qemu-hda nbd: my_nbd_server: 1025
```

On another computer, a second instance to start.

```
Host3 ~ $ qemu-hda nbd: my_nbd_server: 1025
```

With *the-s* or *-snapshot* will not change the image, but in writing the temporary files. For the guest system seems the image is described. These changes will be lost when you shut down the instance. This option slows down the access.

```
Host ~ $ qemu-nbd-s-t-p 1025 Platte.img
```

A complete write protection enabled the *option-r (- read-only)*. Many operating systems can not boot from a read-only media. Therefore, this option is more likely to live CDs or additional memory media is interesting. In the following example, two network block devices are placed simultaneously on different ports. The image *Daten.img* is protected.

```
Host1 ~ $ qemu-nbd-s-t p-1025 drive
Host1 ~ $ qemu-nbd-r-t-p 1026 Daten.img
```

Both Network block devices can be integrated by an instance.

```
Host2 ~ $ qemu-hda nbd: my_nbd_server: 1025 \
    -Hdb nbd: my_nbd_server: 1026
```

Besides images can also CDs, DVDs and floppy disks as a network block device to export. Thus, different installation media provided on a central server.

```
Host1 ~ $ qemu-nbd-t-p 1025 / dev / cdrom
```

The Network Block Device is integrated by an instance on another machine.

```
Host2 ~ $ qemu-cdrom nbd: my_nbd_server: 1025
```

With the *-connect = DEV* Image can directly to a specific network block device assigned. The parameter identifies the device *DEV*.

```
Host ~ $ qemu-nbd - connect = / dev/nbd0 Platte.img
```

It is possible that device or partition to mount.

```
Host ~ $ sudo fdisk-l / dev/nbd0
Host ~ $ sudo mount / dev/nbd0p1 / mnt
```

All options of *qemu-nbd* shows the option *-help*.

```
Host1 ~ $ qemu-nbd-help Usage: qemu-nbd [OPTIONS] FILE QEMU Disk Network Block Device server-p, - port = PORT port to listen on (default '1024')-o, -
```

As a faster alternative to *qemu-nbd* package under Linux, the *nbd-server* or cluster using *the-server* package *gnbd* be used for the. *Nbd-server* provides better protection than access *qemu-nbd*.

```
Host ~ $ sudo apt-get install nbd-server
```

To set an image as a network block device available, the port and the image must be indicated by its path. We recommend using *the-a* option to specify a timeout in seconds. This prevents the unnecessary processes remain in memory during periods of inactivity.

```
Host1 ~ # nbd-server 1025 / path / Platte.img-A 100
```

From a second computer system a guest of the Network Block Device is started.

```
Host2 ~ $ qemu-hda nbd: my_nbd_server: 1025
```

After finishing the instance to check whether the process still of the NBD server is active.

```
Host1 ~ $ ps aux | grep nbd-server [s]
```

Processes are no longer needed to quit.

```
Host1 ~ $ killall nbd-server
```

A complete write protection for the image enables *the-r* option.

```
~ $ Host nbd-server 1025 / path / Platte.img-r-a 100
```

With *the-c* (copy on write) the changes are not written to the image, but in temporary files (*. *diff*). For the guest system seems the image is described. These changes will be lost when you shut down the instance. This option slows down the access.

```
~ $ Host nbd-server 1025 / path / Platte.img a 100-c-
```

All options shows the Man-in page.

```
Host ~ $ man nbd-server
```

To access network block devices can be used in the package-client *nbd-client* or *gnbd* under Linux. This NBD clients do not support booting from a network block device.

```
Host ~ $ sudo apt-get install nbd-client
```

If a network block device with *qemu nbd nbd-server* or made *available*, which can be accessed on *nbd-client* with. For this, the IP address or hostname of the server, the port and the local device is indicated.

```
Host ~ $ sudo nbd-client my_nbd_server 1025 / dev/nbd0
```

An installed guest system can be started with QEMU / KVM.

```
Host ~ $ sudo chmod 777 / dev/nbd0
Host ~ $ qemu-hda / dev/nbd0
```

All options shows the Man-in page.

```
Host ~ $ man nbd-client
```

```
<<< | # # # | >>>
```

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Network_Block_Devices"

This page has been accessed 7713 times. This page was last updated on 27 September 2010 at 18:04 clock changed. Content is available under GNU Free Documentation License 1.2 .

embed images in the host system, mount, umount lomount fdisk

(Link [include [QEMU-KVM-Buch / media / QEMU images in the host system]] to this page)

<<< | # # # | >>> | English

embed images in the host system

Image-files that are used by a host system, the host system must not write (data loss). On Unix / Linux images that contain only one file system, are directly involved through the loopback device. The virtual hard disk has to be present in raw format, partition and the mount command with a known file system to be formatted. First, the image format is determined.

```
Host ~ $ qemu-img info Platte.img
FileFormat: qcow2
```

In this example, the image before the format qcow2. It must be converted.

```
Host ~ $ qemu-img Platte.img-O raw-disk raw.img
```

Then we tested with the *fdisk* partitions of the new image file. The-*/* option shows the partition table. The *option-u* causes that sectors are used as a unit.

```
Host ~ $ sudo fdisk-lu Disk-raw.img
Units = sectors of 1 * 512 = 512 bytes
Device Boot Start End Blocks Id System
Platte * 63 409 247 204 592 raw.img1 + 6 FAT16
```

This image contains only one partition. being calculated, the mount operation, the offset. It is clear from the start sector, here 63, multiplied with the byte sector size, this 512th The offset is thus 32256th As a type, this must be *vfat*.

```
Host ~ # mount plate raw.img / mnt-o loop, offset = 32 256-t vfat
```

Unmount with *umount* is done as usual.

```
Host ~ # umount / mnt
```

In this example, the image of one partition.

```
Host ~ # fdisk-lu Disk-raw.img
Units = sectors of 1 * 512 = 512 bytes
Device Boot Start End Blocks Id System
Platte raw.img1 * 63 32 129 16 033 + 83 Linux
32 130 305 234 136 552 plate-raw.img2 + 82 Linux swap
Platte raw.img3 305 235 10,474,379 5,084,572 + 83 Linux
```

To mount the third partition, an offset is required by $512 * 305,235$ th

```
Host ~ # mount plate raw.img / mnt-o loop, offset = $ ((512 * 305 235))
```

For mounting and manipulating virtual media library is developed *libguestfs* (see http://qemu-buch.de/d/Managementtools/_libguestfs).

<<< | # # # | >>> <http://qemu-buch.de>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_QEMU-Images_im_Host-System_einbinden "

This page has been accessed 9806 times. This page was last updated on 27 September 2010 at 06:09
clock changed. Content is available under GNU Free Documentation License 1.2 .

IT forensics, computer forensics, forensic data recovery, Advanced Forensic Format (AFF), Expert Witness Format (EWF) afflib, aimag, xmount

(Link to this page as [[QEMU-KVM-Buch / media / Forensics Tools]])

<<< | # # # | >>> | English

Tools for the IT Forensics

In the IT forensics digital tracks in computer systems to determine facts are analyzed. Of evidence to backup files are created in storage media. The steps to copy the hard drives are in the *section-to-Virtual Physical* described (see http://qemu-buch.de/d/Speichermedien/_Physical-to-Virtual). These images are analyzed with special programs.

afflib

Website: <http://www.afflib.org>

The package contains *afflib* tools for the *Advanced Forensic Format (AFF)*. The installation on Ubuntu is done with a command line.

```
Host ~ $ sudo apt-get install afflib
```

To as an image format AFF to import a disk, use the command *aimag*. In this example, */ dev / sda* read the name and image *Platte.aff* written.

```
Host ~ $ sudo aimag / dev / sda Platte.aff
```

It can be imported with a command of different media.

```
Host ~ $ sudo aimag / dev / sda Platte1.aff / dev / sdb Platte2.aff
```

For verifying an image of the AFF is the command *afinfo* with *option-v* to apply.

```
~ $ Host-v afinfo Platte.aff
```

To convert an image on the *Advanced Forensic Format* in an image in RAW format is the command used AFCAT.

```
Host ~ $ AFCAT Platte.aff> Platte raw.img
```

For review of conversions, use the command *afcompare*.

```
Host ~ $ afcompare Platte.aff plate raw.img
Platte.aff and plate.raw.img: compare files okay
```

xmount

Download: <https://www.pinguin.lu/index.php>

In computer forensics Images are usually in the Advanced Forensic Format (AFF) or Expert Witness Format (EWF) is stored. To convert these formats is the Linux tool *xmount*. This tool creates virtual images to mount points. These virtual images can, QEMU and other virtualization solutions are used by KVM. *Xmount file system* uses this in *userspace* (FUSE). *Xmount* stands as Debian-/Ubuntu-Paket available. Furthermore, this tool can be compiled from source. To compile the following packages are required (eg Ubuntu):

```
Host ~ $ sudo apt-get install libfuse libfuse2-dev fuse-utils
Host ~ $ sudo apt-get install libcryptfs-dev
```

The sources are downloaded, unpacked and compiled.

```
Host $ mkdir ~ xmount
Host $ cd ~ xmount
Host ~ $ wget - no-check-certificate \
      https://files.pinguin.lu/projects/xmount-0.4.0-src.tar.gz
Host ~ $ tar xzvf xmount-0.4.0-src.tar.gz
Host $ cd ~ xmount-0.4.0
Host ~ $ ./Configure & & make
Host ~ $ sudo make install
```

The list of options on *xmount-h*.

```
Host ~ $ xmount-h
```

xmount can image in the *raw* format (dd), Forensic Format (AFF) and Expert Witness Format (EWF) Advanced Reading. To specify the input format is the *- in*. With *- out* to virtual images in the *RAW* format (dd), VirtualBox and VMware (vmdk) to be generated vdi (. To illustrate the operation of *qcow2* is an image format an image in *raw* format (dd) as shown.

```
Host ~ $ qemu-img info Platte qcow2.img
FileFormat: qcow2
```

not read the format *qcow2* *xmount* As can with this image *qemu-img* convert.

```
Host ~ $ qemu-img convert-O raw-disk-drive-qcow2.img raw.img
```

It is a directory as a mount point for the file system in userspace (FUSE) to create.

```
Host $ mkdir ~ fusemp
```

xmount is with the *- formats* of the call and *- out* in the statement. Then, the name of the to-read image and mount point to be indicated. With the *- cache* file is an overlay set. Thus, the original image is not changed. All changes are written to the specified overlay file.

```
Host ~ $ xmount - cache Platte.ovl - in dd - dd out Platte.img fusemp
```

The mount point is the virtual image now available.

```
Host ~ $ qemu-img info fusemp / Platte.dd image: fusemp / Platte.dd file format: raw virtual size: 230M (241,172,480 bytes) disk size: 0
```

With QEMU or KVM can boot the operating system on this virtual image.

```
Host ~ $ kvm_fusemp / Platte.dd
```

The virtual image can be converted to.

```
Host ~ $ qemu-img convert -O qcow2 fusemp / Platte.dd plate.qcow2.img
```

With *qemu-img* and the parameter *check* can check the image format qcow2.

```
Host ~ $ qemu-img-plate check qcow2.img  
No errors were found on the image.
```

If the virtual image is no longer needed, unhook the FUSE mount.

```
Host ~ $ sudo umount fusemp
```

An image of the *Advanced Forensic Format* (AFF) with the option - as a virtual image shown *aff*.

```
Host ~ $ xmount - in aff - out dd Platte.aff fusemp /
```

Order from existing image files in several formats *Expert Witness* (EWF) as a virtual image in raw format to make one, is the - *ewf adopted*. The names of the EWF files with the wildcard *. E? Addressed. *Xmount* recognizes the files automatically.

```
Host ~ $ xmount - cache Platte.ovl - in ewf - out dd *. E? fusemp
```

```
<<< | # # | >>> http://www.qemu-buch.de
```

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Forensik-Tools"

This page has been accessed 8230 times. This page was last updated on 27 September 2010 at 06:09 clock changed. Content is available under GNU Free Documentation License 1.2 .

Virtual hardware, emulated hardware

(Link [Customize [QEMU-KVM-Buch / Virtual Hardware]] to this page)

<<< | # # # | >>> | English

Virtual Hardware

QEMU and the Kernel-based Virtual Machine support variable adaptation of the virtual hardware.

Processor architectures
Hardware Components
Paravirtualized device drivers (KVM)

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Virtuelle_Hardware_anpassen "

This page has been accessed 9797 times. This page was last updated on 28 October 2009 at 11:52 clock changed. Content is available under GNU Free Documentation License 1.2 .

SMP processor architectures QEMU x86 x86_64 PPC SPARC MIPS pentium3 ARM PowerPC

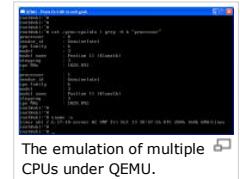
SuperH Cold Fire

(Link to this page as [[QEMU-KVM-Buch / virtual hardware / processors]])

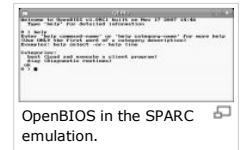
<<< | # # # | >>> | English

Contents

- 1 processor architectures
 - 1.1 x86
 - 2.1 SPARC
 - 1.3 ARM
 - 4.1 MIPS
 - 1.5 Cold Fire
 - 1.6 PowerPC
 - 1.7 SuperH
 - 1.8 ETRAX CRIS
 - 1.9 MicroBlaze
- 2 multi-processor systems
 - 2.1 SMP
 - 2.2 NUMA
- 3 x86 virtualization
 - 3.1 KVM hardware-supported virtualization
 - 3.2 The Accelerator KQEMU



The emulation of multiple CPUs under QEMU.



OpenBIOS in the SPARC emulation.

processor architectures

As a hardware emulator QEMU emulates different architectures. Some Linux distributions do, the package *qemu-kvm-extras* installed.

- PC (x86 or x86_64 processor)
- ISA PC (old PC without PCI bus)
- PREP (PowerPC processor)
- Beige G3 PowerMac (PowerPC processor)
- Mac99 PowerMac (PowerPC processor)
- Sun4m/Sun4c/Sun4d (32-bit Sparc processor)
- Sun4u/Sun4v (64-bit Sparc processor)
- Malta board (32-bit and 64-bit MIPS processors)
- MIPS Magnum (64-bit MIPS processor)
- ARM Integrator / CP (ARM)
- ARM Versatile baseboard (ARM)
- ARM RealView Emulation baseboard (ARM)
- Spitz, Akitia, Borzoi, Terrier and Tosa PDA (PXA270 processor)
- Luminary Micro LM3S811EVB (ARM Cortex-M3)
- Luminary Micro LM3S6965EVB (ARM Cortex-M3)
- Freescale MCF5208EVB (ColdFire V2).
- Arnewsh MCF5206 evaluation board (ColdFire V2).
- Palm Tungsten | E PDA (OMAP310 processor)
- N800 and N810 tablets (OMAP2420 processor)
- MusicPal (MV88W8618 ARM processor)
- Gumstix Connex motherboard and Verdex (PXA255/270).
- Siemens SX1 Smartphone (OMAP310 processor)
- Syborg SVP base model (ARM Cortex-A8)
- AXIS Devboard88 (CRISv32 ETRAX-FS)
- PetaLogix Spartan 3aDSP1800 MMU Ref Design (MicroBlaze).

x86

Most PC operating systems use the x86 architecture. The x86 architecture is based on a CISC instruction set. The acronym CISC stands for *Complex Instruction Set Computing* (computing with complex instruction set). Intel's first 16-bit CPU, which was 8086, the x86 architecture introduced in 1978. Due to the enormous success of the IBM PC and its clones, the x86 architecture in a few years one of the most successful processor architectures. Besides Intel and other manufacturers produce x86 compatible CPUs such as the company AMD. With the 80386, Intel introduced 1985 the first x86 CPU with 32-bit architecture. 2003 began for the x86 64-bit era, this time on the initiative of AMD. The 64-bit standard is called AMD64 and was incorporated under the name EM64T by Intel. For the x86 architecture, the following hardware is emulated:

- SeaBIOS (from QEMU 0.12.0), PC-BIOS from the Bochs project (up QEMU 0.11)
- Plex86/Bochs LGPL VGA BIOS
- Symmetric multiprocessing (SMP) with up to 255 CPUs
- PC-Bus: PCI or ISA system (i440FX Host PCI Bridge and PCI to ISA Bridge PIIX3)
- two PCI IDE interface with support for up to four (virtual) hard drives or CD / DVD-ROMs
- two floppy drives
- CD / DVD drive
- Graphics card (Cirrus CLGD 5446 PCI VGA card or standard VGA graphics card with Bochs VESA BIOS Extension)
- PS / 2 mouse and keyboard
- PCI UHCI USB controller and virtual USB hub
- PCI-/ISA-Netzwerkadapter
- Network cards (Intel E1000 and others)
- Parallel port
- Serial Ports
- Sound card (Creative Sound Blaster 16, ESONIQ AudioPCI ES1370 PCI, Intel 82801AA AC97, Yamaha YM3812 OPL2 Adlib, Gravis Ultrasound GF1, CS4231A)

The 32-bit version of the x86 processor architecture used by QEMU by default.

```
Host ~ $ qemu Platte.img
```

The -M option allows the selection of the emulated machine type (PC bus architecture). If it is omitted or the default *option-M PC*, and a bus today's standard PC with a PCI-emulated. The possible types of machines as the *M-to?*

```
Host ~ $ qemu -M?
Supported machines are:
PC Standard PC (alias of pc-0.13)
pc-0.13 standard PC (default)
pc-0.12 standard PC
pc-0.11 standard PC, qemu 12:11
pc-0.10 standard PC, qemu 12:10
isapc ISA-only PC
Para-virtualized Xen xenpv PC
```

With newer QEMU / KVM versions will change the emulated hardware. QEMU is replaced or the Kernel-based Virtual Machine have an update, some guest systems problems with

the changed hardware. For example, some Microsoft Windows systems must then be reactivated. Thus, the emulated hardware does not change the machine type to the QEMU version is bind. In the following example, the PC machine type set by the QEMU version 0.11.0.

```
Host ~ $ qemu-M pc-0.11 Platte.img
```

With the -M *isapc*, a PC model with the older ISA bus (Industry Standard Architecture emulated. This can be for older operating systems, be useful in the emulation of industrial PCs and embedded systems.

```
Host ~ $ qemu-M Platte.img isapc
```

The *option-cpu* allows you to select the processor type (CPU - Central Processing unit). The possible types of *shows-cpu*?

```
Host ~ $ qemu-cpu?
x86 [n270]
x86 [athlon]
x86 [pentium3]
x86 [pentium2]
x86 [pentium]
x86 [486]
x86 [Core Duo]
x86 [kvm32]
x86 [qemu32]
x86 [kvm64]
x86 [core2duo]
x86 [phenom]
x86 [qemu64]
```

A 486-processor with *cpu-486* emulates.

```
Host ~ $ qemu-cpu Platte.img 486
```

To emulate 64-bit processors used to command *qemu-system-x86_64*. If a 32-bit host with *qemu-system-x86_64* 64-bit architecture emulates below, are neither KVM KQEMU still available.

```
Host ~ $ qemu-system-x86_64 Platte.img
```

The possible types of machines as the *M-to*?

```
Host ~ $ qemu-system-x86_64-M?
Supported machines are:
PC Standard PC (alias of pc-0.13)
pc-0.13 standard PC (default)
pc-0.12 standard PC
pc-0.11 standard PC, qemu 12:11
pc-0.10 standard PC, qemu 12:10
isapc ISA-only PC
Para-virtualized Xen xenpv PC
```

The possible types of CPU as the *CPU at*?

```
Host ~ $ qemu-system-x86_64-cpu?
```

```
x86 Opteron_G3
x86 Opteron_G2
x86 Opteron_G1
x86 Nehalem
x86 Penryn
x86 Conroe
x86 [n270]
x86 [athlon]
x86 [pentium3]
x86 [pentium2]
x86 [pentium]
x86 [486]
x86 [Core Duo]
x86 [kvm32]
x86 [qemu32]
x86 [kvm64]
x86 [core2duo]
x86 [phenom]
x86 [qemu64]
```

Additional CPU the CPU type definitions can be passed by a comma. With the CPU definition + *svm* virtualization techniques are processors with hardware-emulated. That is, the host system can serve as the host system for further KVM instances. This requires that the actual host system via AMD processors. The kernel module *kvm-amd* see is the option with *nested = 1* to enable (<http://qemu-buch.de/d/Installation>). If a virtualization solution within operated another virtualization solution, it is called nesting.

```
Host ~ $ qemu-system-x86_64-cpu Platte.img qemu64, + svm
```

SPARC

SPARC (Scalable Processor Architecture) by Sun Microsystems (now Oracle Corporation) developed in 1985. SPARC processors are mainly found in products from Sun. In addition to SunOS or Solaris can be used on SPARC Linux and BSD variants. In 1995, the original 32-bit architecture to 64-bit extended and marketed under the name of UltraSPARC. The QEMU package is the program *sparc qemu-system* emulation of 32-bit SPARC systems to contain. The following Sun4m architectures are emulated:

- SPARCstation 4
- SPARCstation 5
- SPARCstation 10
- SPARCstation 20
- SPARCserver 600MP
- SPARCstation LX
- SPARCstation Voyager
- sparc classic
- SPARCbook

The emulations of SPARCstation 2 (sun4c) SPARCserver SPARCcenter 1000 and 2000 (sun4d) also possible. The following Sun4m, Sun4c and Sun4d hardware is emulated:

- IOMMU and IO-UNIT
- TCX frame buffer
- Lance (Am7990) Ethernet network card
- NVRAM M48T02/M48T08
- Slave I / O: timer, interrupt controller, Zilog serial ports, keyboard and power / reset logic
- ESP-SCSI controller with hard-disk and CD-ROM Support
- Floppy drive (not for SS-600MP)
- CS4231 sound device (only for SS-5)

The maximum amount of memory (RAM) depends on the type of machine. In SS-5 of the RAM can be a maximum of 256 MB, or 2047 MB. The machine types of *shows-M?*.

```
Host ~ $ qemu-system-sparc-M?
Supported machines are:
SS-5 Sun4m platform, SPARCstation 5 (default)
SS-10 Sun4m platform, SPARCstation 10
SS-600MP Sun4m platform, sparc server 600MP
SS-20 Sun4m platform, SPARCstation 20
Sun4m Voyager platform, SPARCstation Voyager
Sun4m LX platform, SPARCstation LX
SS-4 Sun4m platform, SPARCstation 4
sparc classic Sun4m platform, sparc classic
SPARCbook Sun4m platform, SPARCbook
SS-1000 Sun4d platform, sparc Server 1000
SS-2000 Sun4d platform, SPARCcenter 2000
SS-2 Sun4c platform, SPARCstation 2
Para-virtualized Xen xenpv PC
```

qemu-system-sparc emulates up to 16 CPUs. Linux is limited but the number to a maximum of four CPUs. The possible types of CPU as the *CPU at?*

```
Host ~ $ qemu-system-sparc-cpu? Sparc Fujitsu MB86900 IU FPU 00.08 million 00 million 00 million MMU NWINS 7-swap-mul-div-flush-fsqrt-FMUL Sparc Fujit:
```

For the SPARC emulator is the OpenBIOS (<http://www.openbios.org>) implemented. Goal of OpenBIOS is a complete compatibility with the standard EEE 1275-1994 at the firmware. In QEMU, it is currently not possible to boot Solaris, NetBSD and OpenBSD. The SPARC32-system emulator has these additional options:

```
-G WxHx [xDEPTH]
```

Defines the initial graphics mode (default = 1024x768x8). It can also be set 1024x768x24 mode.

```
String-prom-env
```

This option is given to OpenBIOS variables in the NVRAM. In the following example, the automatic boot process is interrupted and the prompt appears OpenBIOS.

```
Host ~ $ qemu-system-sparc-M sparc classic Platte.img \
1024x768x24-g-prom-env 'auto-boot? = False'
```

Assistance to the OpenBIOS commands to get *help* with.

```
0> help
```

The boot process will start the *boot* command with. Should be CD-ROM is launched, enter *boot cdrom*. Examples of virtual machines with SPARC architecture are discussed in the section guest systems.

The program *qemu-system-sparc64* serves to emulate the following 64-bit SPARC: sun4u (UltraSPARC PC-like), sun4v (T1 PC-like), or generic Niagara Machine (T1) *sparc64*. *Qemu-system-* is still under development and is currently only a few kernel boot. The following sun4u architectures are emulated:

- UltraSPARC IIi APB PCI Bridge
- PCI VGA compatible graphics card with VESA Bochs Extensions
- PS / 2 mouse and keyboard
- M48T59 NVRAM
- PC-compatible serial ports
- two PCI IDE interfaces with hard disk and CD-ROM Support
- Floppy Drive

The possible types of machines as the *M-to?*.

```
Host ~ $ qemu-system-sparc64-M?
Supported machines are:
sun4u platform sun4u (default)
sun4v sun4v platform
Sun4v platform Niagara, Niagara
Para-virtualized Xen xenpv PC
```

The possible types of CPU as the *CPU at?*

```
Host ~ $ qemu-system-sparc64-cpu? Fujitsu Sparc64 Sparc IU 0004000200000000 FPU MMU 00 million 00 million NWINS 4 Fujitsu Sparc64 Sparc III IU 0004000:
```

The SPARC64-system emulator has the same options as the SPARC32 system emulator.

ARM

The ARM architecture in 1983 by the British computer manufacturer Acorn Computers Ltd. started as a development project. Instead of relying on the company's Intel and Motorola processors, they developed their own processor, the ARM (Acorn RISC Machine). Tests showed that these computers at virtually the same clock speed was about eight times faster than the computers of competitors Commodore Amiga and Atari ST. After 1989, the ARM2 to ARM3 had been developed and as more and more companies had expressed an interest in these processors, Acorn, founded in 1990 with Apple and VLSI Technology, the company Advanced RISC Machines Ltd.. (ARM). The ARM RISC architecture follows the concept (Reduced Instruction Set Computing). A RISC instruction set dropped in favor of a lower decoding complexity consistently to complex commands, allowing the individual commands are easier to execute. In addition, the commands for RISC processors are hardwired. Every operation is thus represented by conductors on the processor. Because of their low power ARM processors are used in many embedded systems, such as mobile phones, PDAs and routers are used. In QEMU Package program is the *qemu-system-containing arm*. The possible types of machines as the *M-to?*.

```
Host ~ $ qemu-system-arm-M?
```

The ARM Integrator / CP emulation supports the following hardware:

- ARM926, ARM1026E, ARM946E, ARM1136 and Cortex-A8 CPU
- two PL011 UARTs
- SMC 91c111 Ethernet adapter
- PL110 LCD controller
- PL050 KMI with PS / 2 keyboard and mouse
- PL181 MultiMedia Card Interface: SD Card.

The ARM Versatile baseboard emulation supports the following hardware:

- ARM926, ARM1136 and Cortex-A8 CPU
- LI90 Vectored Interrupt Controller
- four PL011 UARTs
- SMC 91c111 Ethernet adapter
- PL110 LCD controller
- PL050 KMI with PS / 2 keyboard and mouse
- PCI Host Bridge with access to PCI memory. Since no access to PCI IO is possible, some devices (ne2k_pci NIC) is not supported and other devices (rtl8139 NIC) function to apply only if these memory-mapped control register.
- PCI OHCI USB controller.
- LSI53C895A PCI SCSI host bus adapter with hard disk and CD-ROM drives
- PL181 MultiMedia Card Interface: SD Card.

The ARM RealView Emulation baseboard supports the following hardware:

- ARM926, ARM1136, ARM11MPCORE (x4) or Cortex-A8 CPU
- ARM AMBA Generic / Distributed Interrupt Controller
- four PL011 UARTs
- SMC 91c111 Ethernet adapter
- PL110 LCD controller
- PL050 KMI with PS / 2 keyboard and mouse
- PCI Host Bridge
- PCI OHCI USB Controller
- LS153C895A PCI SCSI host bus adapter with hard disk and CD-ROM drives
- PL181 MultiMedia Card Interface: SD Card.

The emulation of the XScale-based clamshell PDA models (Spitz, Akita, Borzoi and Terrier) supports the following hardware:

- Intel PXA270 System-on-chip (ARM V5TE Core)
- NAND Flash Memory
- IBM / Hitachi Microdrive DSCM in a PXA PCMCIA slot - not in "Akita"
- On-Chip OHCI USB Controller
- On-chip LCD controller
- On-chip Real Time Clock
- TI ADS7846 touchscreen controller on SSP bus
- Maxim MAX1111 analog-digital converter to I²C bus
- GPIO-connected keyboard controller and LEDs
- Secure Digital Card connected to the PXA MMC / SD host
- three on-chip UARTs
- WM8750 audio CODEC to I²C bus and I²S bus.

The emulation Palm Tungsten | E PDA (code-named "Cheetah") supports the following hardware:

- Texas Instruments OMAP310 system-on-Chip (ARM 925T core)
- ROM and RAM memory. The ROM firmware image can be *loaded-option-rom* with the.
- On-chip LCD controller
- On-chip Real Time Clock
- TI TSC21021 Touchscreen controller analog-digital converter / audio CODEC connected to micro wire and I²S-Bus
- GPIO-connected matrix keyboard
- Secure Digital Card connected to OMAP MMC / SD host
- three on-chip UARTs

The emulation of the Nokia N800 and N810 Internet Tablet (RX-34, RX-44/48) supports the following hardware:

- Texas Instruments OMAP2420 system-on-chip (ARM 1136 core)
- RAM and non-volatile OneNAND Flash Memories
- Display with EPSON Remote Frame Buffer chip and OMAP-on-chip display controller and a LS041y3 MIPI DBI-C Controller
- TI TSC2301 (in N800) and TI TSC2005 (in N810) touch screen controller with SPI bus
- National Semiconductor LM8323-controlled Qwerty keyboard with I²C bus
- Secure Digital Card connected to OMAP MMC / SD host
- OMAP three on-chip UARTs and On-chip debugging console STI
- a Bluetooth (R) Transceiver and UART HCI-related
- Mentor Graphics' Inventra "dual-role USB controller embedded in a TI TUSB6010 chip - only USB host mode is supported
- TI TMP105 temperature sensor on the I²C bus
- TI TWL92230C power management with an RTC on the I²C bus
- Nokia RETU Tahvo and multi-purpose chip to an RTC on the CBUS

The Luminary Micro Stellaris LM3S811EVB emulation supports the following hardware:

- Cortex-M3 CPU core
- 64K Flash and 8K SRAM
- Timers, UARTs, ADC, and I²C interface
- OSRAM Pictiva 96x16 OLED with SSD0303 controller on the I²C bus.

The Luminary Micro Stellaris LM3S6965EVB emulation supports the following hardware:

- Cortex-M3 CPU core
- 256K Flash and 64K SRAM
- Timers, UARTs, ADC, I²C and SSI interfaces
- OSRAM Pictiva 128x64 OLED with SSD0323 controller connected via SSI.

The Freecom MusicPal Internet Radio emulation supports the following hardware:

- Marvell MV88W8618 ARM core.
- 32 MB RAM, 256 KB SRAM, 8 MB Flash.
- up to two 16550 UART
- MV88W8x8 Ethernet Controller
- MV88W8618 audio controller, WM8750 CODEC and Mixer
- 128 x 64 display with brightness control
- 2 buttons, 2 navigation wheel with button function.

The emulation of the Siemens SX1 models v1 and v2 (default) supports the following hardware:

- Texas Instruments OMAP310 system-on-Chip (ARM 925T core)
- ROM and RAM memory (ROM firmware image can be loaded with *option-pflash*) V1 1 flash 1 Flash 16MB and 32MB Flash 8MB V2 1
- On-chip LCDController
- On-chip Real Time Clock
- Secure Digital Card connected to OMAP MMC / SD host
- three on-chip UARTs

The emulation of Symbian Syborg Virtual Platform supports the following hardware:

- ARM Cortex-A8 CPU
- Interrupt Controller
- Timer
- Real Time Clock
- Keyboard
- Frame Buffer
- Touchscreen
- UARTs

The possible types of CPU as the *CPU at?*

```
Host ~ $ qemu-system-arm-cpu?
```

The ARM emulation supports this additional option:

Semi-hosting

Enables Hosting Semi syscall emulation. In the ARM architecture, the "Angel" is implemented interface. In this case the host system direct access to the host file system is allowed. It is only with trusted host systems used. The problem is the replication of the boot process in this architecture. First, there are significant differences to a normal PC. On the other difficult-licensing provisions, a replica of the hardware and firmware. Booting Linux is possible with the special Linux boot options. Example:

```
Host ~ $ qemu-system-arm-kernel zImage.integrator \
        Initrd-arm_root.img
```

MIPS

The MIPS architecture (Microprocessor without interlocked pipeline stages) is a RISC processor architecture developed in 1981 at Stanford University. MIPS was originally a 32-bit architecture, extended in 1991 with the R4000 to 64 bits. MIPS processors were used by Silicon Graphics in Unix machines. Previously offered other manufacturers, such as Digital Equipment Corporation (DEC) and Siemens and SNI, machines with MIPS processors. MIPS processors are also found commonly in embedded systems (Cisco routers, Sun Cobalt server to RaQ/Qube2, BMW navigation systems, Fritz Box, Satellite Receiver, Dreambox, Konica Minolta and Sony DSLRs and Nintendo game consoles). The QEMU package, the programs are *system-mips qemu*, *qemu-system-mips64*, *qemu-system-mipsel qemu-system-and mips64el* (MIPSEL order is a MIPS-architecture with a different byte) included. The following machines can be emulated.

- Generic ISA PC-like machine, "mips"
- MIPS Malta prototype board "Malta"
- Acer Pica pica61 "(requires 64-bit emulator)
- MIPS emulator pseudo-board mipssim "
- MIPS Magnum R4000 machine "magnum" (requires 64-bit emulator).

The possible types of machines as the *M-to?*:

```
Host ~ $ qemu-system-mips-M?
```

The generic emulation (mips) allows the installation of Debian on a virtual hard disk. The generic emulation supports the following hardware:

- several different MIPS CPUs (default 24Kf)
- PC-compatible serial port
- PC-compatible IDE hard disk
- NE2000 network card

The Malta-emulation supports the following hardware:

- Core board with MIPS CPU and 24Kf Galileo system controller
- PIIX4 PCI / USB / SMBus Controller
- Multi-I / O chip's serial device
- Pnet32 PCI Network Card
- Malta serial FPGA Device
- Cirrus (default) and other PCI VGA graphics cards

The ACER-Pica emulation supports the following hardware:

- MIPS R4000 CPU
- PC-compatible IRQ and DMA controller
- PC keyboard
- IDE Controller

The MIPS emulator pseudo-board emulation (mipssim) supports similar hardware as the generic emulation (mips):

- Several different MIPS CPUs (default 24Kf)
- PC-compatible serial port
- MIPSnet network emulation

The MIPS Magnum R4000 emulation supports the following hardware:

- MIPS R4000 CPU
- PC-compatible IRQ controller
- PC keyboard
- SCSI Controller
- G364 framebuffer

The possible types of CPU as the *CPU at?*:

```
Host ~ $ qemu-system-mips-cpu?
Host ~ $ qemu-system-mips64 cpu?
```

The problem is the replication of the boot process in this architecture. Booting Linux is possible. An example of a virtual machine with MIPS architecture is explained in the section *guest systems*.

Cold Fire

The ColdFire processor architecture is a CISC architecture based on the Motorola 68000 family. These processors are designed specifically for the embedded market. The QEMU package is program *qemu-system-m68k* contain. Boot a uClinux kernel is supported. The possible types of machines as the *M-to?*:

```
Host ~ $ qemu-system-m68k-M?
```

The M5208EVB emulation supports the following hardware:

- MCF5208 ColdFire V2 microprocessor (ISA A + with EMAC).
- three two-on-chip UARTs.
- Fast Ethernet Controller (FEC)

The AN5206 emulation supports the following hardware:

- MCF5206 ColdFire V2 microprocessor.
- Two-on-chip UARTs.

It will support this additional option.

Semi-hosting

Enables Hosting Semi syscall emulation. For the M68K architecture, the "ColdFire GDB" interface is implemented. In this case the host system direct access to the host file system is allowed. It is only with trusted host systems used. The problem is the replication of the boot process in this architecture. Booting Linux is possible. Example:

```
Host ~ $ qemu-system-m68k-kernel vmlinux-2.6.21-uc-Oceanographic
```

PowerPC

The PowerPC (PPC) is a 1991 by a consortium of Apple, IBM and Motorola (now Freescale) specified CPU architecture on RISC-based. PowerPC processors include the use in Apple Macintosh computers, from IBM pSeries (RS/6000) and the IBM BladeCenter JS20, the Motorola power stack machines, the Nintendo GameCube and Wii, in the form of the Cell in the Playstation 3 from Sony and the Xbox 360 from Microsoft, and in many embedded systems. The operating systems Apple Mac OS X, Apple Mac OS versions 7.5 through 9.0, respectively OpenDarwin Darwin and Linux (LinuxPPC) are available for this architecture.

QEMU emulates the following PowerPC hardware:

- UniNorth or Grackle PCI Bridge
- PCI VGA compatible graphics card with VESA Bochs Extensions
- Two PMAC IDE interfaces with hard disk and CD-ROM Support
- NE2000-PCI Adapter
- NVRAM (Non Volatile Random Access Memory)
- VIA CUDA with ADB keyboard and mouse.

QEMU emulates the following hardware PREP:

- PCI Bridge
- PCI VGA compatible graphics card with VESA Bochs Extensions
- 2 PMAC IDE interfaces with hard disk and CD-ROM Support
- Floppy Drive
- NE2000-PCI Adapter
- Serial Port
- PREP NVRAM (Non Volatile Random Access Memory)
- PC compatible keyboard and mouse.

QEMU uses the OpenBIOS for the emulation of the G3Beige and Mac99 PowerMac machines. The QEMU package programs, *the-system-ppc qemu-system-ppc64* and *qemu-system-mentioned ppcemb qemu*. The possible types of machines as the *M-to?*.

```
Host ~ $ qemu-system-ppc-M?
```

The possible types of CPU as the *CPU at?*

```
Host ~ $ qemu-system-ppc-cpu?
```

The following example starts a virtual machine with PowerPC architecture:

```
Host ~ $ qemu-system-ppc-g 800x600 Platte.img
```

qemu-system-ppc has the following additional options:

```
-G WxH [zDEPTH]
```

The *start-g* option is resolution and color depth to the. The default is 800x600x15.

```
String-prom-env
```

The *option-prom-env* is OpenBIOS variables in the NVRAM. Example:

```
Host ~ $ qemu-system-ppc-prom-env 'auto-boot? = False \
-Prom-env 'boot-device = hd: 2, \ yaboot' \
-Prom-env 'boot-args = conf = hd: 2, \ yaboot.conf'
```

The problem is the replication of the boot process in this architecture. Booting Linux is possible.

SuperH

The SuperH processor architecture (SH) is developed by Renesas Technology. This RISC-like processor architecture is very common in Japan. Worldwide SuperH is used in many embedded systems, since this architecture has low power consumption and relatively inexpensive. The HP Jornada series of used SH processors. The QEMU package are the programs *qemu-system-sh4* and *qemu-system-sh4eb* to emulate the 4th Generation (SH-4) included. The possible types of machines as the *M-to?*.

```
Host ~ $ qemu-system-sh4-M?
```

The possible types of CPU as the *CPU at?*

```
Host ~ $ qemu-system-sh4 cpu?
```

The problem is the replication of the boot process in this architecture. Booting Linux is possible. Example:

```
Host ~ $ qemu-system-sh4-M r2d-hda sh-linux.img-kernel zImage \
Zero-serial-serial stdio-Oceanographic
```

ETRAX CRIS

ETRAX CRIS means a family of processors the company Axis Communications. These processors are based on the Code Reduced Instruction Set (CRIS). The QEMU package is program *qemu-system-cris* contain. The possible types of machines as the *M-to?*.

```
Host ~ $ qemu-system-cris-M?
```

The possible types of CPU as the *CPU at?*

```
Host ~ $ qemu-system-cris-cpu?
```

MicroBlaze

The microcontroller is defined as the MicroBlaze soft-core in hardware description languages like VHDL (Very High Speed Integrated Circuit Hardware Description Language) or Verilog HDL. This 32-bit RISC microcontroller is implemented by FPGA (Field Programmable Gate Array) from Xilinx. In addition to the paid version of Microblaze there are open source replicas that can be used on FPGAs from other manufacturers. For example, is the aeMB-MicroBlaze clone under the LGPL license. MicroBlaze is supported by the operating systems µCLinux, Linux and FreeRTOS. The possible types of machines as the *M-to?*.

```
Host ~ $ qemu-system-microblaze-M?
Supported machines are:
petalogix-s3adsp1800 Petalogix Linux for Xilinx Spartan refdesign 3ADSP1800 (default)
```

The possible types of CPU as the *CPU at?*

```
Host ~ $ qemu-system-microblaze-cpu?
```

Multi-processor systems

SMP

A symmetric multiprocessing (SMP) has a multi-processor architecture in which the current processes can be distributed across all available processors. In the asymmetrical multiprocessing other hand, each CPU will be assigned tasks. KVM and QEMU can emulate SMP with up to 255 CPUs. This is configured using *the-smp* followed by a number (1 to 255).

```
Host ~ $ qemu-hda Platte.img-smp 2
```

The QEMU monitor *cpu* informs the command *info* on the CPUs.

```
(Qemu) info cpus
* CPU # 0: pc = 0xc0127365 (halted)
```

```
CPU # 1: pc = 0xc0127365 (halted)
```

The CPU marked with an asterisk is the default CPU. The following command will be # 1 for the default CPU CPU.

```
(Qemu) CPU 1
(Qemu) info cpus
CPU # 0: pc = 0xc0127365 (halted)
* CPU # 1: pc = 0xc0127365 (halted)
```

With the command `cpu_set` can switch on a CPU offline.

```
(Qemu) cpu_set 0 Offline
```

Use the following command, the CPU # 0 online.

```
(Qemu) cpu_set 0 online
```

For x86 architectures can specify the number of cores per socket, the threads per core and the number of sockets. The total number of CPUs can be omitted.

```
Host ~ $ qemu-hda Platte.img-smp cores = 2 threads = 1, = 2 sockets
(Qemu) info cpus
* CPU # 0: pc = 0x00008bdb (halted)
CPU # 1: pc = 0x000ff0a2 (halted)
CPU # 2: pc = 0x000ff0a2 (halted)
CPU # 3: pc = 0x000ff0a2 (halted)
```

Furthermore, the maximum number of hot-pluggable CPUs are defined.

```
Host ~ $ qemu-hda Platte.img \
-Smp cores = 2 threads = 1, Socket = 2, maxcpus = 8
```

NUMA

NUMA (Non-Uniform Memory Architecture) is a memory architecture for multi-processor systems. Each processor has a separate, local memory. Other processors in the system is made possible through a common address space, direct access to this memory (Distributed Shared Memory). NUMA architectures are implemented such as AMD Opteron multi-processor systems on base. In simplified terms, a NUMA node, a memory region in which each byte the same distance (hops) to each CPU. To configure a NUMA node serves *the-node NUMA*. In the following example, a multi-node NUMA system is simulated.

```
Host ~ $ qemu-system-x86_64 disk \
Node-numa, numa-cpu = 0 node, cpus = 1-smp 2
```

The QEMU monitor informs the command `info` about the NUMA *NUMA* node.

```
(Qemu) info numa
2 nodes
node 0 cpus: 0
node size 0: 64 MB
node 1 cpus: 1
node 1 size: 64 MB
```

The COMPLETE syntax is:

```
Numa-node [, mem = size] [, cpu = cpu [-cpu]] [, nodeid = node]
```

The option `mem = size` defines the size of the memory. The default value is 128 MB. If this option is not specified, the resources are shared equally. The option `cpus = cpu [-cpu]` addressed the CPU (s) for configuration. If this option is not specified, the resources are shared equally. The option `nodeid = node` defines the node ID.

x86 virtualization

The KVM hardware-supported virtualization

As described above, the Kernel-based Virtual Machine is a type 2 hypervisor that supports the Full Virtualization. Instead of the command `qemu` is the Kernel-based Virtual Machine often used the `kvm` command. Some distributions call to the `with-system-x86_32 qemu`, `qemu-system-x86_64 qemu` or `kvm-`.

```
Host ~ $ kvm Platte.img
```

If the use of the KVM hardware-assisted virtualization is to be prevented, the `option-no-kvm` specify. When you install Microsoft Windows as guest system, the KVM hardware-assisted virtualization is disabled. Once installed, the KVM hardware-enabled virtualization are supported on all versions of Windows.

```
Host ~ $ kvm-no-kvm Platte.img
```

In the QEMU monitor, the command `info kvm` whether the KVM hardware-assisted virtualization is active. In this case, it was `with-no-kvm` disabled.

```
(Qemu) info kvm
kvm support: disabled
```

If the KVM kernel modules are loaded, the command `info kvm kvm support` the message: from `enabled`.

```
Host ~ $ qemu-monitor stdio
(Qemu) info kvm
kvm support: enabled
```

If a virtualization solution within operated another virtualization solution, it is called *nesting*. The `option-enable-nesting` allows the operation of a KVM KVM instance. For this, the host system must have an AMD processor. It is necessary to emulate the 64-bit CPU. Example:

```
Host ~ $ qemu-system-x86_64 Platte.img \
Qemu64-cpu, + svm-enable-nesting
```

The `option-kvm-shadow-memory` storage size is the KVM MMU-shadowing the predetermined assigned. With the `option-no-kvm-irqchip` disable the KVM kernel mode PIC / IOAPIC / Lapic. The `option-no-kvm-pit` off the KVM kernel mode PIT. The `option-no-kvm-pit-reinjection` off the KVM kernel mode PIT interrupt reinjection.

The Accelerator KQEMU

From QEMU 0.12.0 KQEMU is no longer supported. For instructions on installing QEMU with kqemu-support can be found at the URL http://qemu-buch.de/d/QEMU+KQEMU_unter_Linux.

If the host and guest system available in x86 processor architecture, QEMU can use with the optional accelerator KQEMU the Native Virtualization. It goes KQEMU most commands directly to the real CPU. Only the CPU commands that directly address the hardware are intercepted and replaced with custom routines. For Linux guest systems the best Acceleration with the 2.4 kernel is reached. The 2.6 versions work, but are slower. To enable the full KQEMU virtualization is the `option-kernel-kqemu` apply.

```
Host ~ $ qemu-kernel-kqemu Platte.img
```

The QEMU monitor shows `info kqemu` command the use of the KQEMU.

```
(Qemu) info kqemu
kqemu support: enabled for user and kernel code
```

Another *option-enable-kqemu*. This KQEMU is activated only in the user code.

```
Host ~ $ qemu-Platte.img enable-kqemu
```

The QEMU monitor shows *info kqemu* command the use of the KQEMU.

```
(Qemu) info kqemu
kqemu support: enabled for user code
```

If the use is to be prevented from KQEMU, the *option-no-kqemu* specify. When you install Microsoft Windows as guest system KQEMU should be disabled. After installation, KQEMU be enabled on all versions of Windows.

```
Host ~ $ qemu-no-kqemu Platte.img
```

In the QEMU monitor, the command *info kqemu* whether the accelerator KQEMU is active.

```
(Qemu) info kqemu
kqemu support: disabled
```

In this case KQEMU has been disabled. In the example below KQEMU QEMU compiled without support. *The kernel-kqemu-no-kqemu* and are not available.

```
(Qemu) info kqemu
kqemu support: not compiled
```

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Virtuelle_Hardware/_Prozessoren "

This page has been accessed 16,209 times. This page was last updated on 26 January 2011 at 16:54 clock changed. Content is available under GNU Free Documentation License 1.2 .

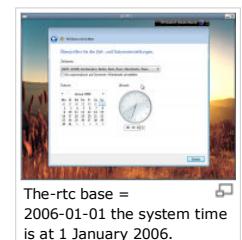
Hardware RAM ACPI BIOS clock local time utc Graphic Keyb USB Sound System Clock USB PCI Ispci Isusb

(Link to this page as [[QEMU-KVM-Buch / Virtual Hardware / RAM bus ACPI Clock Graphic Keyb USB Sound]])

<<< | # # # | >>> | English

Contents

- 1 Hardware components
 - 1.1 RAM
 - ACPI 2.1
 - 1.3 System Clock
 - USB 4.1
 - PCI 1.5
 - 1.6 graphics output, keyboard and mouse
 - 1.7 Sound
 - 1.8 Network Cards
 - 1.9 The Option-device
 - 1:10 Character Devices



Hardware components

Depending on the used processor architecture different hardware components are emulated. By default, standard components can be adjusted. If no default devices are generated, the *option* to apply *nodemults*. The QEMU monitor then shows no *info* command *network* devices on the network.

```
Host ~ $ qemu-Platte.img nodemults
```

In the QEMU monitor, the command displays *info* on *qtree* the device tree.

```
(Qemu) info qtree bus: main-system-type bus system dev: i440FX pcihost-id, "" bus: PCI pci.0 type dev: PIIX3-ide, id "" bus-prop: addr = 01.1 class ID
```

RAM

The amount of RAM (Random Access Memory) defines *the-m* option followed by a number This is the number the amount of RAM in MB. The default value is 128 Optional can be specified using M or G is the size in megabytes or gigabytes. QEMU and KVM support to 2047 MB. For older operating systems, this value is reduced. Newer systems usually require more memory. Here's an example with 16 MB of RAM:

```
Host ~ $ qemu-m 16 Platte.img
```

In the following example the host system on GB RAM is made available.

```
Host ~ $ qemu-m Platte.img 1G
```

Will be allocated on Linux as the host system the host system, a large memory, there can be problems if there is insufficient shared memory. QEMU or KVM then complain about insufficient space in */dev/shm*. It is better to extend this file system. For a permanent configuration) is the file */etc/default/tmpfs* (Ubuntu adapt).

```
Host ~ # umount / dev / shm
Host ~ # mount-t tmpfs-o remount, size = 144m none / dev / shm
```

The adaptation of memory at run time, Linux as guest system with the memory ballooning possible for (see Section *paravirtualized device drivers*).

Information about the memory management to get the QEMU monitor. The *info* command shows active *mem*, virtual memory mapping to the. The *info* command lists *tib* mapping of virtual to physical memory on that.

ACPI

The Advanced Configuration and Power Interface (ACPI) is an open industry standard for power management in computers. The control of the energy management lies entirely with the operating system. ACPI is the default QEMU and KVM. Should the guest operating system any problems, disable ACPI with *-no-acpi*. All versions of Windows 2000 and install different kernel versions, depending on whether ACPI is available. Why can not disable ACPI later if it was enabled during installation.

```
Host ~ $ qemu-no-acpi Platte.img
```

A special ACPI configuration allows the *option-acpitable*. The ACPI table is thereby filled with the specified header fields and contents of the files.

```
-Acpitable [sig = str] [, rev = n] \ [, oem_id = str]
[, Oem_table_id = str] [, oem_rev = n] [, asl_compiler_id = str] \
[, Asl_compiler_rev = n] [, data = file1 [: file2 ]....]
```

System Clock

The System Clock (Real Time Clock) is defined with *the-rtc*. Here's the syntax.

```
-Rtc [base = utc | local time | date] [, clock = host | vm] [, driftfix = none | slew]
```

To configure the system clock is by default the Coordinated Universal Time (UTC). Some operating systems such as DOS and Microsoft Windows versions, require the local time as setting the system time. In Central Europe would mean a difference of one hour in winter (UTC +1 corresponds to the MDGs) and two hours in summer (UTC +2 corresponds to the BST) result. The attitude of the local time option allows *rtc-base = localtime* the. In older versions of QEMU, the local option *local time* with *time-defined*.

```
Host ~ $ qemu-rtc Platte.img base = localtime
```

The-rtc base = date, the system clock when you start the virtual machine. In older versions of QEMU, the start time option *start date* specified. Possible values are 2008-04-17T16:01:21 and 2008-04-17. Below the clock is set to 31.12.1999.

```
Host ~ $ qemu-rtc Platte.img base = 1999-12-31
```

By default, the system of the instance to the system of the host system is provided. With the option *clock = vm* instance is running the system in isolation from the host.

```
Host ~ $ qemu-rtc Platte.img base = utc, clock = vm
```

For problems with time-drift in Microsoft Windows with ACPI HAL option is to apply *slew driftfix* = (only for x86 architectures). It is determined how many timer interrupts are not processed by the Windows guest system and attempts to apply it again. In older versions of QEMU *hack* was for the *option-rtc-td-applied*.

```
Host ~ $ qemu-rtc Platte.img base = localtime driftfix slew =
```

The BIOS on motherboards that are suitable for multi-core processors, is often the High Precision Event Timer (HPET) is implemented. This very precise timer triggers interrupts with a very high temporal resolution (quantization) from. Especially multimedia applications benefit from HPET. New operating systems support the High Precision Event Timer. In the QEMU monitor, the command *info hpet* use of the HPET.

```
(Qemu) info hpet
HPET is enabled by QEMU
```

The option-no-HPET HPET is the support for disabled.

```
Host ~ $ qemu-no-Platte.img hpet
(Qemu) info hpet
HPET is disabled by QEMU
```

USB

The Universal Serial Bus (USB) is a bus system for connecting a computer with accessories. With USB-equipped devices can be connected together during operation (hot plugging). The connected devices and their properties are automatically recognized. Activates the USB emulation with *USB* option. Emulates a PCI UHCI USB controller. USB devices are the-*USBDevice* device configured with. Does the host system via USB support can be accessed on the USB device. the USB *tablet* device with the mouse pointer is in both the guest and the host system as used. The mouse pointer leaves the window of the instance is automatically switched to the hands of the host system. You have to leave so that window without having to enter the key combination [Ctrl] + [Alt]. This option overrides the PS/2-Maus-Emulation.

```
Host ~ $ qemu-usb-Platte.img USBDevice tablet
```

In the QEMU monitor, the command displays *info* on this *usb* USB device.

```
(Qemu) info usb
Device 0.2, speed 12 Mb / s, USB Tablet Product
```

For older operating systems, such as Microsoft Windows 98 CD is indeed a new driver for the USB Device *Tablet* installed from the installation, but the right mouse button does not work. It is a virtual USB device to add the *Mouse*.

```
(Qemu) usb_add mouse
```

the possibilities of *option-USBDevice tablet* addition, the pressure on the pointing device evaluation, if the *option-USBDevice wacom-tablet* is used instead. Here, a virtual Wacom Tablet PenPartner is emulated. This requires the Tslib library.

```
Host ~ $ qemu-Platte.img USBDevice wacom-tablet
```

A virtual USB memory stick to tie with the-*USBDevice disk*: a disk *file*. *File* is an image of a formatted virtual. Here's an example with an image (*usb stick.img*) as a virtual USB memory stick.

```
Host ~ $ qemu-Platte.img USBDevice disk: usb stick.img
```

The QEMU monitor can check with *info usb*, whether the device was added.

```
(Qemu) info usb
Device 0.2, speed 12 Mb / s, USB Product MSD (usb stick.img)
```

USB device is removed with the command *usb_del* this device, with the example of the combination device is in 0.2. Previously, the USB device in the guest system is logging off.

```
(Qemu) usb_del 0.2
```

With the following option, a USB-serial converter is emulated.

```
USBDevice-serial: [VendorID = vendor_id] [, product_id = product_id]: dev
```

The default values for the vendor and product ID are 0403 and 6,001th It is FT232BM FTDI chip and emulates a device connected to the host *dev*. Available character devices are similar to those of the *serial* option. The following example *tmp* output to the file // redirected *My.Log*.

```
Host ~ $ qemu-disk-01.img USBDevice serial:: file: / tmp / My.Log
```

QEMU Manager is in the command information from the following *info usb*.

```
(Qemu) info usb
Device 0.2, speed 12 Mb / s, USB Serial Product QEMU (file: / tmp / My.Log)
```

The *option-USBDevice bt* [: *hci-type*] dongle is a Bluetooth emulated. The possible values for *hci-type* are described in the *Bluetooth* section.

```
Host ~ $ qemu-Platte.img USBDevice bt
```

QEMU Manager is in the command information from the following *info usb*.

```
(Qemu) info usb
Device 0.2, speed 480 Mb / s, Product QEMU BT dongle
```

With the-*net USBDevice*: *options* can be a USB network adapter to emulate, the CDC and RNDIS support the protocols. *Options* for the parameters of the *option-net nic-users*. Currently, this option may not be used with PCI network cards. For example, the user mode network stack is defined with the following options.

```
Host ~ $ qemu-net user Platte.img, vlan = 0-USBDevice net: vlan = 0
```

QEMU Manager is in the command information from the following *info usb*.

```
(Qemu) info usb
Device 0.2, speed 12 Mb / s, Product QEMU USB Network Interface
```

The direct access to a USB device to the host system is still in the experimental stage and force from the QEMU and KVM. Under Linux host system lists the command *lsusb* on the available USB devices.

```
Host ~ $ lsusb
```

QEMU monitor is in the *USB host info* command the host device to check availability of the.

```
(Qemu) info USB host
Device 0.1, speed 480 Mb / s
Class 00: USB device 090c: 1000, USB DISK
```

In this example, host device with the Vender ID *090c* and the product ID *1000* to be accessed on the. This allows the QEMU monitor, the command *usb_add*.

```
(Qemu) usb_add host: 090c: 1000
```

If the access rights in the host system prohibit access, an error message.

```
Warning: could not add USB device host: 090c: 1000
/ Dev/bus/usb/000/001: Permission denied
```

The access rights in Linux using the following line.

```
Host ~ $ sudo chmod-R 777 / dev / bus / usb
Host ~ $ sudo chmod-R 777 / proc / bus / usb
```

Now, in the QEMU monitor, the command will be successfully applied *usb_add*. To access the USB device to the host system may also allow the *option-USBDevice*.

```
Host ~ $ qemu-Platte.img USBDevice host: 090c: 1000
```

To remove the USB device is used in QEMU monitor the command `usb_del`. Previously, the USB device in the guest system is logging off.

```
(Qemu) usb_del host: 090c: 1000
```

PCI

PCI (Peripheral Component Interconnect) is a bus standard for connecting peripheral devices with the processors. Unlike the ISA bus supports PCI, the dynamic configuration of a device without user intervention. Nearly all from 1994, built IBM-compatible PCs equipped with two to seven slots for PCI cards. Also, newer computers from Apple and Sun workstations have a PCI bus. In the PCI slots can accommodate cards of different manufacturers are used, for example, network cards, modems, sound cards and (older or second) graphics cards. The QEMU monitor displays the command `info pci` PCI devices connected to the virtual.

```
(Qemu) info usb
```

The QEMU monitor command adds the `pci_add` hotplug devices launched the host system (Linux) is added. This requires the host system, the kernel modules are loaded and `pci_hotplug aciphp`. In this example, a virtual PCI network interface card (*NIC*) of the type connected to the *e1000* slot 0.

```
(Qemu) pci_add auto nic model = e1000, id = NIC1
OK domain 0, bus 0, slot 4, function 0
```

The following command adds the file *daten.img* a virtual SCSI disk as added.

```
(Qemu) pci_add auto storage file = daten.img, if = scsi
OK domain 0, bus 0, slot 5, function 0
```

The syntax of the command is `pci_add`:

```
(Qemu) pci_add auto [[:<domain>:] [:<bus>:] <slot> nic | storage \
[[:Vlan = n] [, macaddr = addr] [, model = type]] \
[File = file] [, if = type] [, bus = n]
```

Connected virtual PCI devices with the command `pci_del` away. It is at a minimum, the slot number indicated. With this command, the device in PCI slot 4 is removed.

```
(Qemu) pci_del 4
```

The syntax of the command is `pci_del`:

```
(Qemu) pci_del [[:<domain>:] [:<bus>:] <slot>
```

Still in development, access to PCI devices on the host system. There are still many limitations.

- The host system is a Linux kernel, version 2.6.28 is required.
- The host system must be no drivers on the PCI device access.
- The PCI device may be no IRQ sharing with another device.
- The kernel module *raw_io* is necessary.

The-host = pcidevice busadresse binds a PCI device of a host system. The possible PCI bus addresses lists the command *lspci*.

```
Host ~ $ lspci
06:00.0 ...
```

The following command sets the device to the PCI bus address of *06:00.0* at guests' disposal.

```
Host ~ $ qemu-Platte.img pcidevice host = 06:00.0
```

The syntax of *option-pcidevice* is:

```
Pcidevice-host = bus: dev.func [, dma = none] [, name = string]
```

Dma = none is specified, no DMA translation. Otherwise *IOMMU* is the default value applied. The parameter is the *string* used for logging output.

graphics output, keyboard and mouse

The type of graphics card is from QEMU 0.10.1 *vga option-type* defined. The following types are available:

std

Simulate a standard VGA card with VESA Bochs Extensions. Does the host system, the VESA 2.0 VBE extensions, this option for higher resolutions (> = 1280x1024x16) are selected.

cirrus

It emulates a CL-GD5446 PCI VGA card. This option is set as the default. All versions starting from Microsoft Windows 95 recognize this graphics card. For optimal performance in the host and guest system in each set a color depth of 16 bits.

vmware

Simulates a VMware SVGA II adapter compatible. This option can be used for guest systems with installed video drivers, the VMware Tools.

xenfb

Xen emulates a frame buffer. For a Linux guest system module, the corresponding command *modprobe* with *xenfb* loaded.

none

Disables the VGA card.

The following example enables the standard VGA card with VESA BIOS Extensions 2.0.

```
Host ~ $ qemu Platte.img std-vga
```

If the screen output of the host system in text mode can be applied *to-curses* option. Instead of the VGA output with the DirectMedia Layer (SDL), the VGA output through the Curses / Ncurses interface. In graphical mode at this option, no output.

```
Host ~ $ qemu-curses Platte.img
```

The-activated sdl Simple DirectMedia Layer (SDL) for graphics output.

```
Host ~ $ qemu Platte.img-vnc: 1-sdl
```

An instance in full screen mode starts the *option-fullscreen*. To switch between full screen and windowed mode is the key combination [Ctrl] + [Alt] + [F].

```
Host ~ $ qemu-full-screen Platte.img
```

The-Oceanographic disabled the graphical output and QEMU or KVM to be command-line tools. The emulated serial port is redirected it to the console. This can be controlled using special keyboard shortcuts. An overview of the shortcut shows the key combination [Ctrl] + [A] and [H]. The instance is terminated, for example, [Ctrl] + [A] and [X]. What kind of desktop computers not make much sense, can be interesting for server applications and kernel developers. The lack of interaction of the graphical output can be substituted for another communication. In initial tests, the *snapshot* option recommended. Is no communication with the host system possible, the instance with [Ctrl] + [A] and [X] be ended

without harm to the host system.

```
Host ~ $ qemu-snapshot-Platte.img Oceanographic
```

The *option-no-frame* off the window frame of the instance. *With-no-quits* to disable the window close button (top right). Terminating the instance is the command to *quit* the QEMU monitor still possible. With *the-old-grave +*, the [Ctrl] + [Alt] + [Shift] instead of [Ctrl] Alt to release the mouse used [. The *option-ctrl-grave*] the right [Ctrl] button instead of [Ctrl] + [Alt] to release the mouse uses.

Normally, the keyboard layout is not specified. *The-k* option is only required if in recognition of the PC keyboard layouts are problems. The available layouts are *ar, de-ch, it, fr, fr-ca, hu, yes, mk, no, pt-br, sv, da, en-gb, et, fr, fr-ch, is, lt, nl, pl, ru, th, en, en-us; fi, fr-be, hr, it, lv, nl-be, pt, sl, tr*. Default is *en-us*.

```
Host ~ $ qemu-vnc Platte.img 1-k de
```

The default is for the x86 architecture emulates a PS / 2 keyboard. The *option-USBDevice keyboard* is a USB keyboard emulates.

```
Host ~ $ qemu-usb-Platte.img USBDevice keyboard
```

The default is for the x86 architecture emulates a PS / 2 mouse. The QEMU monitor informs the command *info* on *mice*, the mice.

```
(Qemu) info mice
* Mouse # 0: QEMU PS / 2 Mouse
```

If a USB mouse to be emulated *mouse* is the *option-USBDevice* apply.

```
Host ~ $ qemu-usb-Platte.img USBDevice keyboard-mouse USBDevice
```

The QEMU monitor lists the command *info* on the *usb* USB devices.

```
(Qemu) info usb
Device 0.1, speed 12 Mb / s, Product QEMU USB Keyboard
Device 0.0, speed 12 Mb / s, USB Hub Product QEMU
Device 0.0, speed 12 Mb / s, Product QEMU USB Mouse
```

The *option-chardev msmouse, id = id* be the guest system, the events of a Microsoft mouse to emulate.

```
Host ~ $ qemu-Platte.img chardev msmouse, id = Speedy-Gonzalez
```

The QEMU monitor informs the command *info* on *mice*, the mice.

```
(Qemu) info Mouse mice # 0: QEMU * Microsoft Mouse Mouse # 1: QEMU PS / 2 Mouse
```

Sound

To enable the emulation of one or more sound cards or the PC speaker is the *option-soundhw*. The emulation of the *Adlib* sound card, *Gus* and *cs4231a* are only available when compiling the *option-audio-card-list* the desired sound card was listed with. The following command lists the available sound hardware:

```
Host ~ $ qemu-soundhw?
pcspk PC speaker
sb16 Creative Sound Blaster 16
cs4231a CS4231A
adlib Yamaha YM3812 (OPL2)
gus Gravis Ultrasound GF1
AC97 Intel 82801AA AC97 audio
es1370 ENSONIQ AudioPCI ES1370
```

Several sound cards are separated by commas.

```
Host ~ $ qemu-Platte.img soundhw pcspk, sb16
```

In emulation of the sound card Intel 82801AA AC97 audio under Linux as guest system requires the kernel module *i810_audio* the following option: *modprobe i810_audio clocking = 48000th* The *all* parameter activates all supported sound cards.

```
Host ~ $ qemu-Platte.img soundhw all
```

Information to the audio subsystem shows the *option-audio-help*.

```
Host ~ $ qemu-audio-help
```

By orders of the QEMU monitor can record the sound output of the host system. The following command starts the recording and writes it to the specified file:

```
(Qemu) wavcapture mitschnitt01.wav
```

Other parameters control the sample rate, Samplebits and the number of channels. Information about active recordings, the command *info capture*.

```
(Qemu) info capture
```

The first column shows the index number. The first active recording gets the number 0 To stop recording, *stop* the *capture* command from the index number is entered followed.

```
(Qemu) stop capture 0
```

QEMU was with the *--card-list = adlib, gus, cs4231a* compiled *audio*, *CS4231A* are Yamaha YM3812 sound cards, Gravis Ultrasound GF1 and Crystal emulated. For the emulation of the sound card Gravis Ultrasound (GUS) is GUSeMu (<http://www.deinmeister.de/gusemu/>) was used. By default, the CIS uses IRQ7. This IRQ is also claimed by parallel ports. disabled to avoid conflicts to the parallel port.

```
Host ~ $ qemu-Platte.img soundhw gus-parallel none
```

Alternatively, to set IRQ to the CIS.

```
Host ~ $ qemu-device Platte.img gus, irq = 5
```

NIC

To be emulated NIC (Network Interface Card - NIC) *option-net nic* configured with. The exact description is in the *Network options*.

Option-device

A more flexible configuration of hardware components enables *the-device* option. The syntax is:

```
-Device driver [, prop [= value ][,...]]
```

Adds the device *driver* added.

```
prop = value
```

Defines the properties of the device.

A list of possible devices is *device-?* Issued. This command lists the devices for the x86 architecture (64-bit).

```
Host ~ $ qemu-device?
name "virtio-balloon-pci" PCI bus name "virtio-serial-pci", PCI bus, aka "virtio-serial" name "virtio-net-pci" PCI bus name "virtio-blk-pci", bus PCI na
A standard VGA card with sample-device emulates the VGA. This corresponds to the option-vga std.
Host ~ $ qemu-device Platte.img VGA
With-device driver? Get a list of possible properties for the specified device.
```

```
Host ~ $ qemu Platte.img VGA-device?
VGA.bios-offset = hex32
VGA.bios-size = hex32
```

The QEMU monitor command adds the *device_add* a device run time the virtual machine to add. The options are similar to those of *device*. It should be added only hot-pluggable devices during operation, otherwise there is a crash of the host system. In the following example, the image is *daten.img* as a USB storage added. This USB is activated.

```
Host ~ $ qemu-usb Platte.img
```

The QEMU monitor the following commands are called:

```
(Qemu) drive_add 0 id = mein_usb_drive, if = none, file = daten.img
(Qemu) device_add usb-storage, id = mein_usb_storage, drive = mein_usb_drive
```

The *info* command shows *usb* USB device to it.

```
(Qemu) info usb
Device 0.2, speed 12 Mb / s, Product QEMU USB MSD
```

Removed, this USB device with the command *device_del*.

```
(Qemu) device_del mein_usb_storage
```

Character Devices

Character device (character device) is transmitted only one character or one byte at a time (serial data transmission). The data is usually unbuffered, so transferred immediately. This distinguishes them from the block devices. These transmit the data in data blocks. The *option-chardev* devices are character-defined. Depending on the type of devices (*backend*) options are certain to apply. Any device with a string *id* must be assigned. This is used for unique identification. The general syntax is:

```
Chardev backend-id = id [, options]
```

The Device *Device zero* is an empty created. It does not send data and discarding received data.

```
Host ~ $ qemu-Platte.img chardev null, id = nothing
```

The QEMU monitor *info* lists the command *chardev* character devices on.

```
(Qemu) info chardev
nothing: filename = null
Monitor: filename = vc
serial0: filename = vc
parallel0: filename = vc
```

The *stdio* Device connection is a standard input and output QEMU instance on the. After the start of the instance you change the key combination [Ctrl] + [Alt] + [3] to the standard input and output. This option is not host systems running Microsoft Windows.

```
Host ~ $ qemu-Platte.img chardev stdio, id = hans
```

The device connects *vc* text console to QEMU ago. At that text console with the key combination [Ctrl] + switch [Alt] + [2].

```
Host ~ $ qemu-snapshot-Platte.img chardev vc, id = knut
```

Other ways *the-chardev* be listed in the Annex.

<<< | # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Virtuelle_Hardware/_Bus_RAM_ACPI_Clock_Graphic_Keyb_USB_Sound "

This page has been accessed 15,508 times. This page was last updated on 17 January 2011 at 17:25 clock changed. Content is available under GNU Free Documentation License 1.2 .

Paravirtualized device drivers **virtio, virtio_pci, virtio_ring, virtio_net, virtio_blk, virtio_balloon, Memory Ballooning**

(Link to this page as [[QEMU-KVM-Buch / Virtual Hardware / paravirtualized device drivers]])

<<< | # # # | >>> | English

Contents

A paravirtualized device drivers for the guest systems

- 1.1 driver for Microsoft Windows
- 2.1 Virtio modules for Linux
 - 1.2.1 paravirtualized network cards
 - 1.2.2 paravirtualized block devices
 - 1.2.3 Memory Ballooning

paravirtualized device drivers for the guest systems

The Kernel-based Virtual Machine support usually is not the paravirtualization. The use of paravirtualized device drivers is possible. The advantages of these drivers are lower overhead and higher performance. Only the availability of hardware enhancements such as IOMMU make the use of paravirtualized device drivers redundant.

Drivers for Microsoft Windows

Download: <http://www.linux-kvm.com/content/tip-how-setup-windows-guest-paravirtual-network-drivers>

For Microsoft Windows 2000/XP is a special network driver is available. This is to be installed in the host system. In the *options-net nic virtio* is the parameter *model = add*.

```
Host ~ $ kvm-net nic Platte.img, model = virtio ....
```

Virtio modules for Linux

Virtio is a paravirtualization interface that only one set of host drivers are developed. These drivers can be used by multiple virtualization solutions. This para virtual catalyzed device drivers are available for Linux guest systems with kernel version 2.6.25.

paravirtualized network cards

In the guest system, the following modules are loaded: *virtio, virtio_pci, virtio_ring, virtio_net* and *virtio_blk*. This requires *the-net nic* parameter *model = virtio* be added.

```
Host ~ $ kvm-net nic Platte.img, model = virtio-net tap, ifname = TAP01
```

paravirtualized block devices

Para-virtualized block devices with the *option-drive ... if = virtio* enabled.

```
Host ~ $ kvm-drive file = Platte.img, if = virtio, boot = on
```

In the host system through testing as to whether these modules were loaded.

```
Guest ~ $ lsmod | grep virtio virtio_blk 16 392 17 668 0 0 virtio_pci virtio_ring 12 928 14 336 3 1 virtio_pci virtio virtio_blk, virtio_net, virtio_p...
```

When using Virtio block devices should be in the Linux guest, the ext3 file system can be used. The XFS file system is not recommended because it operates an intense write-caching. In case of sudden failure of the virtual machine can cause problems.

Memory Ballooning

With memory ballooning is defined as the change in size of memory in the current host system. QEMU and the Kernel-based Virtual Machine support the Memory Ballooning only for Linux guests. be installed on the host system has a kernel of version 2.6.27. To activate the memory ballooning with *the-balloon virtio [, addr = st]*. *Addr* can optionally with the PCI device address can be specified.

```
Host ~ $ qemu-balloon Platte.img virtio
```

On the host system *virtio_balloon* the kernel module to load.

```
Host ~ # modprobe virtio_balloon
```

The QEMU monitor identified and change the amount of RAM. Increasing the memory of *the-m* option with a predetermined value is not possible. In the following example, the amount of memory of 512 MB of RAM to 256 MB of RAM is reduced.

```
(Qemu) info balloon
balloon: actual = 512
(Qemu) balloon 256
(Qemu) info balloon
balloon: actual = 256
```

The amount of RAM in the host system can use the command to control *free*.

```
Guest ~ $ free
total ...
Mem: 252 292 ...
```

The *option-none balloon* Ballooning is the memory disabled.

```
Host ~ $ qemu-balloon Platte.img none
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Virtuelle_Hardware/_Paravirtualisierte_Ger%C3%A4tetreiber "

This page has been accessed 21,296 times. This page was last updated on 27 September 2010 at 06:16 clock changed. Content is available under GNU Free Documentation License 1.2 .

Network options, networking

(Link to this page as [[QEMU-KVM-Buch / network options]])

<<< | # # # | >>> | English

Network Options

Configuring Virtual Networks

Network Services

Network Protocol Analysis

Bluetooth

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Netzwerkoptionen "

This page has been accessed 10,923 times. This page was last updated on 26 Amended in February 2010 at 12:11 clock. Content is available under GNU Free Documentation License 1.2 .

KVM QEMU Internet network bridging bridge nat, virtual networks, Slirp TUN / TAP, bridge-utils

(Link to this page as [[QEMU-KVM-Buch / Network options / configure Virtual Network]])

<<< | # # # | >>> | English

Contents

- 1 Virtual Network
 - 1.1 Network Card
 - 2.1 User mode network stack
 - 1.2.1 Port-redirect from the host system (hostfwd)
 - 1.2.2 Redirects from the host system (guestfwd)
 - 1.3 TUN / TAP network interfaces
 - 1.4 More instances with network sockets
 - 1.4.1 TCP Socket
 - 1.4.2 UDP multicast socket
 - 1.5 Bridging and Switching
 - 1.5.1 bridge-utils
 - 1.5.2 Proxy ARP bridging with parprouted
 - VDE 1.5.3 Switch
 - 1.6 The option-netdev

Virtual Networks

QEMU and the Kernel-based Virtual Machine support network connections between the host and the host system. Running multiple instances are also network connections between the host systems possible.

NIC

A virtual NIC (Network Interface Card) is *the-net nic* configured with.

```
Host ~ $ qemu-net nic Platte.img
```

The QEMU monitor *info* informs the command *network* of VLANs (Virtual Local Area Network) of the instance and the associated devices. In this example, the instance is a virtual network (VLAN 0) combined. Given a virtual network interface card (NE2000 PCI) is made with the MAC address 52:54:00:12:34:56 available.

```
(Qemu) info network
VLAN 0 devices:
user.0: net = 10.0.2.0, restricted = n
e1000.0: model = e1000, macaddr = 52:54:00:12:34:56
```

The *option-net nic* is enabled by default and can be omitted.

```
Host ~ $ qemu Platte.img
```

In the QEMU monitor, the command, the virtual network cable for the specified network card removed *set_link (off)* or connected (*on*). The following command pulls out the virtual network cable.

```
(Qemu) set_link e1000.0 off
```

The connection is restored.

```
(Qemu) set_link e1000.0 on
```

If no network card to emulate, is *none* *the-net* use.

```
Host ~ $ qemu-net none Platte.img
```

The syntax for *the-net nic* is as follows:

```
-Net nic [, vlan = n] [, macaddr mac =] [, model = type] \
[, Name = str] [, addr = str] [, vectors = v]
```

The parameter *vlan = n* when using multiple network cards to provide and define the VLAN. The default value is 0 With the *model* parameters can be adjusted network cards. Depending on the processor architecture of different network interface cards are emulated. All available network *cards-net nic, model =?* Listed with.

```
Host ~ $ qemu-net nic, model =?
qemu: Supported NIC models: ne2k_pci, i82551, i82557b, i82559er, rtl18139, e1000, PCnet, virtio
```

The Kernel-based Virtual Machine are partially listed different network cards.

```
Host ~ $ kvm-net nic Platte.img, model =?
qemu: Supported ISA NICs: ne2k_isa
qemu: Supported PCI NICs: i82551 i82557b i82559er ne2k_pci PCnet rtl18139, e1000 virtio
```



In this example, the card is emulated RTL8139.

```
Host ~ $ qemu-net Platte.img nic, model = rtl8139
```

The parameter *macaddr* is a MAC address (Media Access Control), which uniquely identifies your device on the network is used for.

```
Host ~ $ qemu-net nic Platte.img, macaddr = 00:80: AD: 3B: 3E: 4F
```

Some operating systems manage tables in which MAC addresses to network interfaces are assigned. a MAC address is changed, no longer works, the associated network card. Then the mapping table is updated. In *Ubuntu*, identify the MAC addresses in the file and SuSE in /etc/udev/rules.d/30-net_persistent_names.rules adapt etc/udev/rules.d/70-persistent-net.rules. Older Ubuntu versions store the MAC addresses in /etc/iftab. This file can be deleted.

The network card can be better addressed with the *name* parameter to be named = *st*. The QEMU monitor *info* with this name is displayed *network*.

```
Host ~ $ qemu-net nic Platte.img, name = NIC1
(Qemu) info network
VLAN 0 devices:
NIC1: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

With the parameters and *vectors addr = str = v* can be at PCI network cards, the PCI device address and the number of MSI-X vectors (Message Signaled Interrupts) set. This specification has currently no effect on virtio cards. With *vectors X = 0* MSI disabled.

In addition to one or more virtual network cards you need a virtual network infrastructure (cables, hubs, switches, bridges, routers, firewalls, network services,...).

User mode network stack

A virtual network that is implemented without administrator rights in user mode of the host computer is easy to configure. If QEMU or Kernel-based Virtual Machine without entering a network option or with the *options-net nic-net user* called default, is a virtual network (10.0.2.0), firewall and DHCP server (10.0.2.2) generates. The DHCP server assigns the host system automatically assigns an IP address from 10.0.2.15. The routing of which was provided by QEMU gateway (10.0.2.2) is defined. A network configuration is, therefore, in most operating systems do not need and access from the host system to the outside is now available. That is, if the host system has access to the Internet, has also the host system to automatically access the Internet.

```
Host ~ $ qemu-net nic-net Platte.img user
```

If neither for *nor-net nic-net user* additional parameters are necessary, they can be omitted, since these options are enabled by default.

```
Host ~ $ qemu Platte.img
```

If more than one network card connected to a network, each with is with *the-net nic, vlan = n* the VLAN for the network card and the user mode network stack state. Here's an example with two network cards and two VLANs.

```
Host ~ $ qemu-net nic Platte.img, vlan = 0-net user, vlan = 0 \
          -Net nic, vlan = 1-net user, vlan = 1
```

The QEMU monitor informs the command *info* about *network* VLANs to the instance and the associated devices.

```
(Qemu) info network
VLAN 0 devices:
  user.0: net = 10.0.2.0, restricted = n
  ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
VLAN 1 devices:
  user.1: net = 10.0.2.0, restricted = n
  ne2k_pci.1: model = ne2k_pci, macaddr = 52:54:00:12:34:57
```

The QEMU monitor VLANs can be removed with the command *host_net_remove*. In this example, the VLAN 1 will be removed.

```
(Qemu) host_net_remove a user.1
(Qemu) info network
VLAN 0 devices:
  user.0: net = 10.0.2.0, restricted = n
  ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
VLAN 1 devices:
  ne2k_pci.1: model = ne2k_pci, macaddr = 52:54:00:12:34:57
```

The command *host_net_add user* can add VLANs.

```
(Qemu) host_net_add user vlan = 1
(Qemu) info network
VLAN 0 devices:
  user.0: net = 10.0.2.0, restricted = n
  ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
VLAN 1 devices:
  ne2k_pci.1: model = ne2k_pci, macaddr = 52:54:00:12:34:57
  user.1: net = 10.0.2.0, restricted = n
```

For better distinction can be summed up VLANs. In this example, the VLANs and *user.0 user.1 VLANextern* and *VLANintern* but not named.

```
Host ~ $ qemu Platte.img \
-Net nic, vlan = 0-net user, vlan = 0, = VLANextern \ name
-Net nic, vlan = 1-net user, vlan = 1, name = VLANintern
```

The QEMU monitor these names are displayed to the *network* with command *info*.

```
(Qemu) info network
VLAN 0 devices:
VLANextern: net = 10.0.2.0, restricted = n
ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
VLAN 1 devices:
VLANintern: net = 10.0.2.0, restricted = n
ne2k_pci.1: model = ne2k_pci, macaddr = 52:54:00:12:34:57
```

By default, the host system, the network segment 10.0.2.0 / 8 is provided. *With-net user, net = addr [/ mask]*, the network address and optional network mask to be adjusted. In this example, the host system on the network segment 192.168.1.0/24 is given. DHCP gives the host system's IP address 192.168.1.15.

```
Host ~ $ qemu-net nic-Platte.img net user, net = 192.168.1.0/24
(Qemu) info network
VLAN 0 devices:
user.0: net = 192.168.1.0, restricted = n
ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

The host system is responsive to the host system via the second IP address (xxx2) in the host network. In the default network segment (10.0.2.0), this is the IP address 10.0.2.2. This IP address can be tested with *ping*. Furthermore, one can log on the host system via this IP address. *With the-net user, host = addr* can be the IP address of the host system to adapt.

```
Host ~ $ qemu-net nic-net Platte.img user, host = 10.0.2.7
```

The firewall blocks all incoming instance of connections to this virtual network. Therefore it can not be accessed from outside the virtual machine. The host system is protected. By the way, QEMU and KVM block the Internet Control Message Protocol (ICMP) to the outside. That is, a ping from the host system is not functioning to the Internet. If the host system are completely isolated, is *the-net user, y = restrict* use. Log in to the host system via the internal IP address is not possible.

```
Host ~ $ qemu-net nic-net Platte.img user, restrict = y
```

In the QEMU monitor, the command *info* to *network*, whether activated *restrict*.

```
(Qemu) info network
VLAN 0 devices:
user.0: net = 10.0.2.0, restricted = y
ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

The-net user, restrict = y does not affect the forward rules described below (*-net user*, and *hostfwd-net user guestfwd*). Other parameters *the-net user* are described in the section of *network services*.

Port-redirect from the host system (hostfwd)

Since the VLAN is protected by the built-in firewall, the host system from the outside can not be reached. In order to access certain services of the host system, you must have the appropriate ports open in the firewall. This can be in the user mode network stack *the-net user =... hostfwd* be configured to redirect port-by. In each case, a port of the host system to a mapped port of the host system. This means that is open on the host system, a port that is a port of the host system. The use of privileged ports (0 - 1024) of the host system is only the user *root*. The following example can be on port 12345 of the host system to SSH (port 22) on the host system. This must of course run on the host system on SSH server.

```
Host ~ $ qemu-net nic-Platte.img net user hostfwd = tcp::: 12345 -: 22
```

With an SSH client to log on here 12 345 port of the host system. *The-p* option defines the port. It is also a login from another computer on the network in this guest-mode operation. There must be *local host*'s IP address or host name of the host system can be specified. On Microsoft Windows, a log with putty or with Cygwin is possible. The Cygwin commands are like their counterparts on Unix / Linux.

```
Host ~ $ ssh-p 12 345 root @ localhost
```

With SCP or WinSCP on Unix and Linux Microsoft Windows files with the host system to be replaced. The following example uses the file */etc/fstab* of the host system (Linux) on the host system (Linux) *scp* copied.

```
Host ~ $ scp-P 12 345 root @ localhost: / etc / fstab file.
```

To a Remote Desktop Protocol (see # http://qemu-buch.de/d/Anhang/_Nützliche_Tools_Remote/Desktop_Protocol a guest system running Microsoft Windows log-on) can do is to unlock each of the Port 3389th

```
Host ~ $ qemu-net nic-Platte.img net user hostfwd = tcp::: 3389 -: 3389
```

With *rdesktop* or another RDP client connects you to the guest.

```
Host ~ $ rdesktop-g 800x600-k de localhost
```

You can configure multiple port redirects. If you want, for example, the HTTP (port 80) and HTTPS (port 443) of a guest system running on the webserver effect is available, the following options. On the host system in the Web browser using the URL http://localhost:8080

the HTTP port and using the URL `https://localhost: 8081` port of the host system continues to be the HTTPS.

```
Host ~ $ qemu-net nic Platte.img \
-Net user, hostfwd = tcp:: 8080 - 80, hostfwd = tcp:: 8081 - 443
```

The QEMU monitor informs the command `info` on the `Usenet` defined forward rules.

```
(Qemu) info Usenet
VLAN 0 (user.0):
Protocol [State] FD Source Address Port Port Dest.Address recvq SendQ
TCP [HOST_FORWARD] 6 * 8080 10.0.2.15 80 0 0
TCP [HOST_FORWARD] 5 * 8081 10.0.2.15 443 0 0
```

Furthermore, the QEMU monitor rules with the new forward command will be added `hostfwd_add`. The following example can be on port 12345 of the host system to SSH (port 22) on the host system.

```
(Qemu) hostfwd_add tcp:: 12345 -: 22
```

The command `hostfwd_remove` rules Forward deleted.

```
(Qemu) hostfwd_remove tcp:: 12345
```

The general syntax for the Port Redirect by `option-net user, hostfwd` is as follows.

```
-Net user, hostfwd = [tcp | udp]: [hostaddr]: host port [guestaddr]: guest-port
```

If neither `tcp` or `udp` as the connection type specified, the TCP protocol is used. By specifying `hostaddr`, the forward rule defined in the host network interface bound to be one. If not specified `guestaddr`, whose address is the first value from the internal DHCP server default IP assigned (xxx15).

The general syntax for the command `hostfwd_add` QEMU monitor is in the following. In addition to the parameters `of-net user, hostfw VLAN ID` can be specified. This is more virtual network cards required at (see-`net nic, vlan = n`).

```
(Qemu) hostfwd_add [vlan_id name] \
[Tcp | udp]: [hostaddr]: host port [guestaddr]: guest-port
```

The general syntax for the command `hostfwd_remove` is:

```
(Qemu) hostfwd_remove [vlan_id name] [tcp | udp]: [hostaddr]: host port
```

In older QEMU-/KVM-Versionen is for `port-to` redirect the use `redir` option.

Redirects from the host system (guestfwd)

With `the-net user, guestfwd = [tcp]: server: port-dev` is a TCP connection from the host system to the IP address (`server`) and port (`port`) on the character device (`dev`) of the host system passed. This option can be specified multiple times. In the following example, `port` is a TCP connection from the host system to the IP address 10.0.2.1 and the `stdio` redirected to 80th Will host system, an HTTP request to the IP address sent in, this request appears to `stdio`.

```
Host ~ $ qemu Platte.img \
-Net nic-net user, guestfwd = tcp :10.0.2.1:80-stdio
GET / HTTP/1.0
User-Agent: Wget/1.11.4
Accept: * / *
Host: 10.0.2.1
Connection: Keep-Alive
```

/ TUN TAP-Network-Interfaces

The user mode network stack has some limitations. As the guest system is in a different network than the host system and behind a firewall, a complete access to the host system is not possible. This gives the host and guest system, a common network, an additional network interface in the host system is required. This additional network interface will do a / TAP adapter allows. TUN and TAP are virtual network kernel drivers that simulate network devices through software. DO simulates a point-to-point network device, whereas TAP is an Ethernet device. Only by / TUN TAP adapter of the host system the host system receives a full TCP / IP connection. The Linux kernel version 2.1.60 offers from TUN / TAP support. It is necessary to ensure that the device `/dev/net/do` for the user under the QEMU or KVM runs accessible.

```
Host ~ $ sudo chmod go + rw / dev / net / do
```

What is needed is the script `/etc/qemu-ifup`, the call to QEMU with `the-net tap` in the DO / TAP adapter assigns an IP address. This file is as following:

```
# / Bin / sh
sudo-p "Password for $ 0:" / sbin / ifconfig $ 1 10.0.2.100
```

The script in the specified IP address must be on the same network segment as the IP address of the host system are located, but may not match an existing IP address. Configures interface is a network with `ifconfig`. This requires root privileges, which are awarded with the `sudo` command in the script. Furthermore, the script `/etc/qemu-ifdown` necessary. You lay it with the following contents:

```
# / Bin / sh
sudo-p "Password for $ 0:" / sbin / ifconfig $ 1 down
```

QEMU or KVM is *the-net nic-net tap*, and its parameters and start with.

```
Host ~ $ qemu-hda Platte.img-net nic, vlan = 0-net tap, vlan = 0, ifname = tap0
Password for / etc / qemu-ifup: *****
```

The host system receives a TAP-device with the IP address 10.0.2.100.

```
Host ~ $ ifconfig eth0
tap0  Protocol: Ethernet ...
      inet addr: 10.0.2.100 ...
```

The QEMU monitor lists the command *info* this *network* configuration.

```
(Qemu) info network
VLAN 0 devices:
  tap.0: ifname = tap0 script = / etc / qemu-ifup, down script = / etc / qemu-ifdown
  ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

In the QEMU monitor, the command *host_net_remove* this interface removed.

```
(Qemu) host_net_remove 0 tap.0
Password for / etc / qemu-ifdown: *****
(Qemu) info network
VLAN 0 devices:
  ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

To activate a TUN / TAP network interface can in QEMU monitor the command used to *tap host_net_add*. After this command, the parameters as in *tap-net* state.

```
(Qemu) host_net_add tap vlan = 0, ifname = tap0
Password for / etc / qemu-ifup: *****
(Qemu) info network
VLAN 0 devices:
  ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
  tap.0: ifname = tap0 script = / etc / qemu-ifup, down script = / etc / qemu-ifdown
```

This is the full syntax for *the-net tap*:

```
-Net tap [, vlan = n] [, name = name] [, fd = h] [, ifname = name] \
[, Script = file] [, script = dfile down] [, sndbuf = nbytes]
```

Here, the TAP network interface of the host system with the VLAN *N*. It is the script *file* (default = */etc/qemu-ifup*) for the activation and the script *dfile* (Default = */etc/qemu-ifdown*) used for the deactivation. Where the performance of the scripts are prevented, is *no script = script* or *down* to set *no* =. With *fd* is a handle to an existing TAP network interface to be specified. With the option *sndbuf* buffer is the size of the send-limited. The default value of *sndbuf* = 1048576 can be deactivated with *sndbuf* = 0.

Under Linux, it is possible for the program *tunctl* with TAP interfaces or manually generate. This is necessary to move virtual machines from other virtualization solutions networked. This package is the *uml-utilities* needed.

```
Host ~ $ sudo apt-get install uml-utilities
```

The following command sets TAP interface *tap0* to the user and makes the *hans* of the owner.

```
Host ~ $ sudo tunctl -t tap0 -u hans
```

The command *ifconfig* interface on this list.

```
Host ~ $ ifconfig eth0
```

Microsoft Windows versions do not offer / TUN TAP support. This will only be established by the installation of the TUN / TAP adapter OpenVPN. A tutorial is located at the URL http://qemu-buch.de/d/QEMU_unter_Microsoft_Windows.

Multiple instances Sockets network with

With *the-net socket*, it is possible to connect with each other several instances. Is also a link desired to the outside, a virtual machine can be configured as a router. Communication is done via sockets. A socket is a bi-directional software interface for inter-process or network communication.

TCP socket

A variation to allow multiple guest systems communicate with each other is connected through a TCP socket (Transmission Control Protocol). The configuration is similar to the configuration of the UDP multicast socket. If *instance-net socket* with the specified parameter *list* one for, instance waits for incoming connections on the specified port. In the second instance, connects you to *connect* with the first instance. Here is an example of waiting on connection instance.

```
Host ~ $ qemu-disk 01.img-net nic, macaddr = 52:54:00:12:34:56 \
-Net socket, listen =: 1234
```

The second example is an instance that connects to the first instance.

```
Host ~ $ qemu-disk 02.img-net nic, macaddr = 52:54:00:12:34:57 \
```

```
-Net socket, connect = 127.0.0.1:1234
```

In both instances, each in QEMU monitor console with *info network* availability of the socket to be tested. In the instance *list* displays the following output:

```
(Qemu) info network VLAN 0 devices: NE2000 PCI macaddr = 52:54:00:12:34:56 socket: connection from 127.0.0.1:1472
```

the instance to *connect* to this information is displayed.

```
(Qemu) info network
VLAN 0 devices:
socket: connect to 127.0.0.1:1234
ne2000 macaddr pci = 52:54:00:12:34:57
```

Now the IP addresses in the host systems are configured. In these examples, Linux is the guest. The first system receives the IP address 10.0.2.16. This is the command *ifconfig* defined with.

```
Guest 1 ~ # ifconfig eth0 10.0.2.16
```

The second host system receives the IP address 10.0.2.17.

```
Guest 2 ~ # ifconfig eth0 10.0.2.17
```

Here is the complete syntax for generating a TCP socket:

```
-Net socket [, vlan = n] [, name = name] [, fd = h] \
[, Listen = [host]: port] [, connect = host: port]
```

These options connect to the VLAN to another VLAN *n* over a TCP socket connection. Indicated *listen*, instance waits for incoming connections on the specified port (*host* is optional.) In the second instance, connects you to *connect* with the first instance. With *fd* is a handle of an existing TCP socket can be specified.

UDP multicast socket

Another option to connect multiple host systems with each other, is a VLAN via UDP multicast socket. Via multicast to be sent in TCP / IP network data to many recipients at the same time. This is a special multicast address in the range 224.0.0.0 to 239.255.255.255. The protocol UDP (User Datagram Protocol). There is a minimal, connectionless network protocol that is part of the transport layer of the Internet protocol suite. To a UDP multicast socket to connect instances, each of the startup options *nic-net* showing the MAC address of the virtual network card and *net-socket* joint, indicating the multicast address and port to use. In this example, the multicast address 230.0.0.1 is addressed with the port 1234.

```
Host ~ $ qemu-disk 01.img-net nic, macaddr = 52:54:00:12:34:56 \
-Net socket, mcast = 230.0.0.1:1234
```

The second instance is started with a different MAC address.

```
Host ~ $ qemu-disk 02.img-net nic, macaddr = 52:54:00:12:34:57 \
-Net socket, mcast = 230.0.0.1:1234
```

It can have instances are called.

```
Host ~ $ qemu-disk 03.img-net nic, macaddr = 52:54:00:12:34:58 \
-Net socket, mcast = 230.0.0.1:1234
```

In the instances in each case shows the QEMU monitor, the command *info socket* to the *network* used. For example, here the first instance:

```
(Qemu) info network
VLAN 0 devices:
socket.0: socket: mcast = 230.0.0.1:1234
ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

Now the IP addresses in the host systems are configured. In these examples, Linux is the guest. The first system receives the IP address 10.0.2.16. This is the command *ifconfig* defined with.

```
Guest 1 ~ # ifconfig eth0 10.0.2.16
```

The second host system receives the IP address 10.0.2.17.

```
Guest 2 ~ # ifconfig eth0 10.0.2.17
```

The third host system receives the IP address 10.0.2.18.

```
Guest 3 ~ # ifconfig eth0 10.0.2.18
```

In the current instance does it with the command *host_net_remove* remove a socket.

```
(Qemu) host_net_remove 0 socket.0
(Qemu) info network
VLAN 0 devices:
ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

The command generates `host_net_add socket` is a socket. The parameters correspond to those of `net-socket option` to generate a UDP multicast socket.

```
(Qemu) host_net_add socket mcast = 230.0.0.1:1234
(Qemu) info network
VLAN 0 devices:
    ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
    socket.0: socket: mcast = 230.0.0.1:1234
```

Here is the complete syntax for generating a UDP multicast socket:

```
-Net socket [, vlan = n] [, name = name] [, fd = h] [, mcast = Maddr: port]
```

This generates one VLAN, the `n`, which together with other instances of the same multicast address and port can be used `Maddr: port`. Multiple instances can run on different hosts and use the same bus. The multicast Supportt is compatible with User Mode Linux (`eth n = mcast`). With `fd` is a handle to an existing multicast UDP sockets can be specified.

Bridging and Switching

Bridging refers to a technique for coupling two networks, usually LAN segments on the layer 2 of the OSI model. This technique can be used to crosslink virtual machines. This makes it possible to make the virtual machines have full network access. On the virtual machines can also be accessed from outside the host system. When switching, the incoming Ethernet frame is analyzed. Here, the MAC addresses of the sender and receiver in the MAC table will be saved. Thus, packages can be quickly routed to the switch port on which depends the recipient. Since the wiring of the ports can be changed, old entries in the MAC table will be deleted regularly. There are Layer 2 and Layer 3 switches. The latter have usually have management functions. Switches are also known as intelligent hubs and how they work is a bridge very similar. Bridges have usually had only two ports. Switch between four, however, feature, 12 to 48 ports. Switches control more functions than Bridges. To put it simply: Each switch is a bridge, but not every bridge is a switch. However, there are bridges that can also link protocols such as Token Ring and Ethernet (MAC bridge or LLC-Bridge). This is not possible using switches.

bridge-utils

Virtual Bridges are Unix / Linux with the package `bridge-utils` under realized. The Bridging the package `bridge-utils` does not work with wireless devices. For WLAN Devices Bridging the proxy ARP `parproutd` apply to (see below). The installation of the package `bridge-utils` is Debian / Ubuntu the command line:

```
Host ~ $ sudo apt-get install bridge-utils
```

To view existing bridges, and its configuration, the command is `brctl show`.

```
Host ~ $ sudo-i
Host ~ # brctl show
bridge name bridge id STP enabled interfaces
pan0 8000.000000000000 no
```

A Bridge to delete the command `brctl delbr`.

```
Host ~ # brctl delbr pan0
```

To create a new bridge option is to apply `addr`. In the following example, the applied Bridge's name `bri0`.

```
Host ~ # brctl addbr bri0
```

With `brctl show` you check whether the bridge was built.

```
Host ~ # brctl show
bridge name bridge id STP enabled interfaces
bri0 8000.000000000000 no
```

Since only one bridge is used, the Spanning Tree Protocol (STP) is not necessary and disable. The Spanning Tree Protocol is used to avoid redundant network paths (loops) on the LAN.

```
Host ~ # brctl stp off bri0
Host ~ # exit
Host ~ $
```

In this example, the TUN / TAP network interfaces of two virtual machines to scale Bridge can be connected. This allows the virtual machine to the network of the host computer are connected. The address of this network is 192.168.1.0. First, two instances are TUN / TAP network interfaces restart. For the clear need to address the MAC addresses can be different. Otherwise there will be dropouts.

```
Host ~ $ qemu-net nic Platte1.img, macaddr = 52:54:00:12:34:56-net tap
Password for / etc / qemu-ifup: *****
Host ~ $ qemu-net nic Platte2.img, macaddr = 52:54:00:12:34:57-net tap
Password for / etc / qemu-ifup: *****
```

The host system interfaces, the virtual network and `tap1` created `tap0`. This can be checked `ifconfig` command with the. The IP addresses of the virtual network interfaces must be deleted.

```
Host sudo ifconfig tap0 0.0.0.0 up
Host ~ $ sudo ifconfig tap1 0.0.0.0 up
```

The virtual network interfaces are then assigned to the Bridge *bri0*.

```
Host ~ $ sudo brctl addif bri0 tap0
Host ~ $ sudo brctl addif bri0 tap1
```

This assignment is with the command to control *brctl show*.

```
Host ~ $ sudo brctl show
bridge name bridge id STP enabled interfaces
bri0 8000.76674ec80764 no tap0
                                         tap1
```

The bridge gets a free IP address in the network of the host system.

```
Host ~ $ sudo ifconfig bri0 192.168.1.220 up
```

The availability of these IP address will be another computer on that network with *ping* tested.

```
Host 2 ~ # ping 192.168.1.220
```

The virtual machines, free IP addresses of the host network are assigned. Runs on Linux guest systems, the following commands are used.

```
Guest 1 ~ # ifconfig eth0 192.168.1.221 netmask 255.255.255.0 up
Guest 2 ~ # ifconfig eth0 192.168.1.222 netmask 255.255.255.0 up
```

Thus the network card of the host system can be connected to the bridge, his IP address is deleted and then assigned to the Bridge. The host is then no longer available at the old IP address.

```
Host ~ $ sudo ifconfig eth0 0.0.0.0 up & & brctl addif bri0 eth0
```

In order for the bridge even after a restart is available, the appropriate network configuration file is to be adapted. For Debian / Ubuntu, this is the file */etc/network/interfaces*. In this example, the previous entries are deleted to *eth0*. The bridge and thus the computer in this example are accessible via the IP address 192.168.1.220.

```
auto eth0
iface eth0 inet manual
auto bri0
bri0 iface inet static
    address 192.168.1.220
    network 192.168.1.0
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.10
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
    bridge_maxage 12
    bridge_stp off
```

If an address by DHCP server should assign to this configuration is used:

```
auto eth0
iface eth0 inet manual
auto bri0
iface inet dhcp bri0
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
    bridge_maxage 12
    bridge_stp off
```

Then, the network service is restarted.

```
Host ~ $ sudo / etc / init.d / networking restart
```

With many desktop distributions that network configuration is done using network manager. For the configuration shown here, this must be removed. On Debian / Ubuntu this is done with one line.

```
Host ~ $ sudo apt-get remove network-manager
```

In Fedora several steps to uninstall the network manager are necessary:

```
Host ~ $ su -
Host ~ # yum remove NetworkManager
Host ~ # chkconfig network on
Host ~ # echo "DEVICE = eth0">> / etc/sysconfig/network-scripts/ifcfg-eth0
Host ~ # echo "ONBOOT = yes">>> / etc/sysconfig/network-scripts/ifcfg-eth0
Host ~ # system-config-network
```

proxy ARP bridging with parprouted

Another possibility is to use the proxy ARP bridging. Unlike standard bridging, proxy ARP allows the bridging and connecting via a wireless connection. Package is needed to the *parprouted*.

```
Host ~ $ sudo apt-get install parprouted
```

To bridge between the interfaces *eth0* and *tap0* to create one that serves the following command.

```
Host ~ $ sudo parprouted eth0 tap0
```

VDE Switch

Switch to VDE (Virtual Distributed Ethernet) supports the development of complex virtual network connections. This package is the *vde2* necessary.

```
Host ~ $ sudo apt-get install vde2
```

This package contains the program *vde_switch* for creating virtual switches. After his call, and another pressing the Return key will get a prompt. The *help* command shows commands at all. The *shutdown* command stops the virtual switch.

```
Host ~ $ sudo vde_switch [Return] vde vde $ $ help shutdown
```

With the - *daemon* is started as a daemon *vde_switch*. The option - defines the socket *sock*, and with the - *mgmt* is the connection to the management of the virtual switch created.

```
Host ~ $ sudo vde_switch - daemon - sock / tmp / myswitch \
- Mgmt / tmp / myvde.mgmt
```

unix term with the program can manage the switch itself. The command *port vde / print* lists the active ports. With the key combination [Ctrl] + [d] to log off.

```
Host ~ $ sudo unix term / tmp / myvde.mgmt
vde $ port / print
0000 DATA END WITH '. "
.
1000 Success
vde $ [Ctrl] + [d]
Host ~ $
```

Was QEMU / KVM support for VDE compiled with, the *option-net vde* available. With this command, the instance with the applied virtual switch is connected.

```
Host ~ $ qemu-net nic-net Platte.img vde, sock = / tmp / myswitch
```

Was QEMU / KVM VDE compiled without support for, the program included in the package *vde2 vdeq* be used. It is before the command or enter *kvm qemu*.

```
Host vdeq ~ $ qemu-net nic-net Platte.img vde, sock = / tmp / myswitch
```

As a control can be the active virtual switch ports on the list. In this example, a virtual machine is connected to the switch.

```
Host ~ $ sudo unix term / tmp / myvde.mgmt
vde $ port / print
0000 DATA END WITH '. "
Port 0001 untagged_vlan = 0000 ACTIVE - Unnamed ALLOCATABLE
IN: 75 bytes 7103 pkts
OUT: 0 bytes 0 pkts
- Endpoint ID 0003 module unix prog: QEMU user = knut PID = 1740
SOCK = / tmp/myswitch/.01740-00000
```

In the host system, this is for Linux, an IP address is configured.

```
Host ~ # ifconfig eth0 10.0.0.16 netmask 255.255.255.0 up
```

A virtual router can also be set up with *slirpvde*. *Slirpvde* is also part of the package *vde2* and requires no administrative rights. If not yet done is to create a virtual switch earlier.

```
Host ~ $ sudo vde_switch - daemon - sock / tmp / myswitch \
- Mgmt / tmp / myvde.mgmt
```

Now is called *slirpvde*. The option *dhcp-server* is a DHCP-enabled. This gives the virtual machines on virtual switch access to the outside.

```
Host ~ $ sudo slirpvde - daemon - sock / tmp / dhcp-myswitch
```

It can start a virtual machine and connected to the virtual switch.

```
Host ~ $ qemu-net nic-net Platte.img vde, sock = / tmp / myswitch
```

If a second instance of the virtual switch are connected, ensure that they used a different MAC address.

```
Host ~ $ qemu-drive \ zwei.img
-Net nic, macaddr = 00:80:AD:3B:3E:4F-net vde, sock = / tmp / myswitch
```

As a control to list the active ports of the virtual switch.

```
Host ~ $ sudo unix term / tmp / myvde.mgmt
vde $ port / print
```

Here is the complete syntax for *the-net vde*.

```
-Net vde [, vlan = n] [, name = name] [, sock = socketpath] \
[, Port = n] [, group = groupname] [mode = octalmode]
```

The VLAN port *n* is the *n* switches connected with a VDE. This virtual switch is installed on the host system and waits for incoming connections on the socket *socketpath*. With the group *groupname* and the mode *octalmode* it is possible to obtain ownership and access rights to adapt.

The-netdev

QEMU with 0.12.0, the *option* to define virtual networks introduced *netdev*. Depending on the type of virtual network (*user, tap, vde, socket*) are certain parameters apply. must be a string that uniquely identifies *id* assigned to this case. In this example, *with-net nic* a network interface card (*e1000*) and with the *user-mode* network stack defined *netdev*.

```
Host ~ $ qemu Platte.img \
-Net nic, model = e1000, netdev = netdev net0-user, id = net0
```

The definition of the network card with the *option-device* also take place.

```
Host ~ $ qemu Platte.img \
E1000-device, netdev = netdev net0-user, id = net0
```

<<< | # # # | >>>

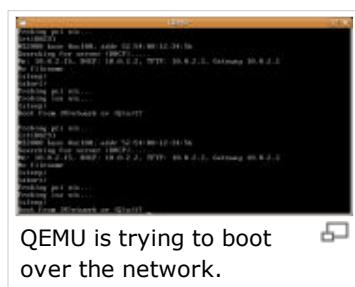
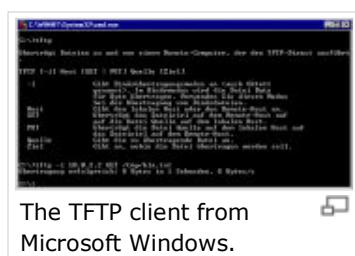
Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Netzwerkoptionen/_Virtuelle_Netzwerke_konfigurieren "

This page has been accessed 39,629 times. This page was last updated on 7 January 2011 at 12:31 clock changed. Content is available under GNU Free Documentation License 1.2 .

dhcp, dns, VNC, SASL, x509, TLS, TFTP Server, PXE boot, gPXE, NBD, BKO, certtool

(Link to this page as [[QEMU-KVM-Buch / Network Options / Network Services]])

<<< | # # # | >>> | English



Contents

- 1 network services
 - 1.1 DHCP and DNS
 - 2.1 Samba
 - 1.3 TFTP
 - PXE 4.1
 - 1.5 VNC

Network Services

DHCP and DNS

The built-in QEMU / KVM DHCP (Dynamic Host Configuration Protocol) supports automatic network configuration of the guest systems in the user mode network stack. This is with almost any network-enabled guest operating systems for an immediate network connection possible to the outside. If the *options-net nic-net user* specified, the virtual network (10.0.2.0), an internal DHCP server (10.0.2.2) and DNS server (10.0.2.3) enabled. The guest systems are assigned IP addresses from 10.0.2.15.

```
Host ~ $ qemu-net nic-net Platte.img user
```

As the *options-net nic-net user* by default are enabled, they can be omitted.

```
Host ~ $ qemu Platte.img
```

-Net user, hostname = host, a system with the integrated DHCP server assigned DNS name for the host-defined. In this example, the IP address of the host system's DNS entry *knut*.

```
Host ~ $ qemu-net nic-Platte.img net user, hostname = knut
```

The option *addr = dhcpstart* defines the first address of the IP range, DHCP server can assign the. This area includes 16 IP addresses. By default, this region lies between xxx16 xxx31 and. For example, with *dhcpstart = 10.0.2.50* set by DHCP to the host system an IP address from 10.0.2.50.

```
Host ~ $ qemu-net nic-Platte.img net user dhcpstart = 10.0.2.50
```

The option *addr = dns* defines the visible to the host system IP address of the integrated DNS server. By default, this is the third IP address in the guest network, for example xxx3. In this example, the DNS server is assigned the IP address 10.0.2.7. This address is communicated to the host system via DHCP.

```
Host ~ $ qemu-net nic-net Platte.img user, dns = 10.0.2.7
```

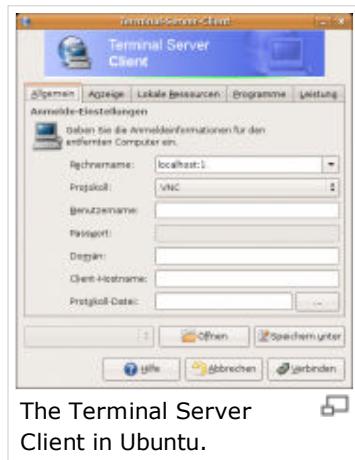
In addition to automatic configuration via DHCP is in the host system's network configuration done manually. As a gateway, the IP address can be set 10.0.2.2. The IP address of the virtual network adapter can be configured in the network segment 10.0.2.0 from 10.0.2.15. Also, other or additional DNS servers are entered. For example, to configure a DNS server according to the instructions at <http://www.opennicproject.org/migrate> in the host system, the top-level *domains. glue. indy. geek. zero. oss* and *. OpenNIC parody* of the project are resolved.

Samba

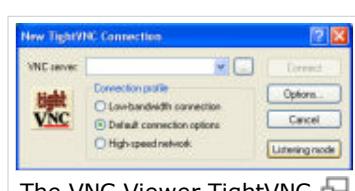
Samba is a free software suite that makes the Server Message Block protocol (SMB) for Unix systems available. Which means that Samba will be Windows shares on Unix / Linux produced. QEMU and KVM allow Samba to exchange data between the host and guest systems. Was on the host system is already installed Samba server, and configured, the host system which shares the URL `\ \ 10.0.2.4 \` accessed. Since neither QEMU KVM still own Samba Server is on the host system is a Samba server installed. On Debian or Ubuntu installation is done with a command line.

```
Host ~ $ sudo apt-get install samba
```

Furthermore, a directory on the host system is required as a share. The Samba server must have read / write permissions for that directory. With *the-net user, smb = directory* of the Samba server (`/usr/sbin/smbd`) with its own configuration started. Thus, the Samba server can be accessed, the user must run under the QEMU or



The Terminal Server Client in Ubuntu.



The VNC Viewer TightVNC.

KVM is supposed to have permission to start the Samba server. A variant is QEMU with `sudo` access.

```
Host ~ $ sudo qemu-net nic-Platte.img net user smb = / tmp /
Password: *****
```

The Samba server has the IP address 10.0.2.4. In Microsoft Windows guest system one uses the File Manager to this share to (*Tools menu, Map Network drive: \ \ 10.0.2.4 \ qemu*). If you prefer to use a host name for the Samba server can assign the IP address of a name. This is the host system, depending on the version of Windows in one of the files C: \ WINNT \ SYSTEM32 \ DRIVERS \ ETC \ LMHOSTS, C: \ WINDOWS \ LMHOSTS or C: \ WINNT \ SYSTEM32 \ DRIVERS \ ETC \ HOSTS the following to add a row.

```
10.0.2.4 smbserver
```

Optionally, the IP address of the Samba server with the option to change `smbserver`. In this example, the Samba server, the address 10.0.2.7 is assigned.

```
Host ~ $ sudo qemu Platte.img \
-Net nic-net user, smb = / tmp /, smbserver = 10.0.2.7
```

TFTP

The Trivial File Transfer Protocol (TFTP) is a simple file transfer protocol. It only supports reading and writing files. Not present many features of the powerful FTP such as rights issue, the ads are existing files or user authentication. Motivation for developing TFTP was the loading of operating systems or configurations on the network. This is the connection-oriented TCP and FTP based on it way too cluttered. TFTP, however, is deliberately kept simple. The built-in QEMU / KVM TFTP server (IP address 10.0.2.2) allows a simple way to put the files in the host system the host system to download. be configured on the host system to be installed and a TFTP client in binary mode. Enables the TFTP server with `the-net user`, followed by the path to `tftp` download directory. For this, the Unix convention is applied. If the host is a Microsoft Windows system, instead of *C: Users \ User01* spelling */ Users/Users01* apply **. A switch of drives is not possible. The following example is used to download files in directory `/ tmp` on the host system.

```
Host ~ $ qemu-net nic-Platte.img net user tftp = / tmp /
```

If not present, the host system or a TFTP client to be installed. If the host system on Debian derivative, the package is to use `tftp`.

```
Guest ~ $ sudo apt-get install tftp
```

Downloading a file from the TFTP host is done with these commands:

```
Guest ~ $ tftp 10.0.2.2
tftp> binary
tftp> get foo.txt
tftp> quit
```

On Microsoft Windows NT, 2000 and XP already have a TFTP client from the DOS command prompt is available.

```
Guest C: \> tftp-i 10.0.2.2 GET foo.txt
```

PXE

The Preboot Execution Environment (PXE) allows a network-based boot process. PXE uses the TCP / IP, UDP, DHCP, TFTP and the concepts GUID / UUID, UNDI (Universal Network Device Interface) and a client-side software extension of defined APIs. With a PXE-enabled computer, it is possible to boot over the network from a boot server. This is necessary if installed on a computer without adding storage media is an operating system or a thin client system should be established. The network boot process is initiated by an installed on the network card BIOS. The boot process works as follows: The firmware provides a DHCP request to determine the PXE boot server. After receiving a reply, they contacted the appropriate boot server to get from him the path to download the NBP (Network Bootstrap Program) submitted. This boot loader is then loaded into memory and executed. QEMU was in version 0.12.0 of the network Bootloader gPXE (<http://etherboot.org>) integrated. PXE gPXE supplemented with additional functions. For example, only supports FTP, HTTP, and iSCSI.

The `option-boot n | o | p` booting over the network will allow. With *n* being the first Network card, *o* the 2nd NIC and the 3rd *p* Network card is addressed. For booting over the network the appropriate PXE ROM images of the network cards are required.

```
Host ~ $ qemu-boot n
No valid PXE rom found for network device
```

In that case, QEMU no PXE ROM image. To load PXE ROM images on the URLs <http://etherboot.org> and <http://rom-o-matic.net> down. In Linux they are in the `/usr/share/kvm` or `/usr/share/qemu` copy to. Elegant is the installation of the parcel.

```
Host ~ $ sudo apt-get install kvm-pxe
```

These are distinguished by the PXE ROM images only in the directory `/usr/share/kvm` or copied.

```
Host ~ $ dpkg -L kvm-pxe | grep . "Bin"
/_usr/share/kvm/pxe-e1000.bin
/_usr/share/kvm/pxe-ne2k_pci.bin
/_usr/share/kvm/pxe-pcnet.bin
/_usr/share/kvm/pxe-rtl18139.bin
/_usr/share/kvm/pxe-virtio.bin
```

QEMU investigated the PXE ROM images under `/usr/share/qemu`. You have to path with the `option-rom` specify the.

```
Host ~ $ qemu-boot-option-rom /usr/share/kvm/pxe-ne2k_pci.bin n
```

It is better in the `/usr/share/qemu` to the appropriate PXE ROM images in the `/usr/share/kvm` symlinks to point to.

```
Host $ cd /usr/share/kvm
Host ~ $ sudo for i in `ls` * pxe; do \
    ln-s /usr/share/kvm/$i /usr/share/qemu/; done
Host ~ $ qemu-boot n
```

With PXE, it is possible to install operating systems on the network. In this example it is Ubuntu. First, a directory must be created. Then the netboot archive download and extract.

```
Host ~ $ sudo mkdir netboot
Host $ cd ~ / netboot
Host ~ $ sudo wget \
http://archive.ubuntu.com/ubuntu/dists/karmic/main/installer-i386/current/images/netboot/netboot
Host ~ $ sudo tar xzvf netboot.tar.gz
```

In any directory is generated for the virtual disk to install QEMU or KVM and then run with the appropriate options. The `boot-file` parameter defines the BOOTP file. Installing Ubuntu is listed in the *guest systems* (http://qemu-buch.de/d/Gast-Systeme/_x86-Architektur/_Linux described).

```
Host ~ $ qemu-img create-f qcow2 ubuntu.img 10G
Host ~ $ qemu-hda-boot ubuntu.img n \
-Net nic-net user = tftp / netboot / boot file = pxelinux.0
```

The options for booting over the network, it is still possible diskless clients for the Linux Terminal Server Project (<http://www.ltsp.org>) to emulate. For example, use the Linux distribution Eubuntu an LTSP server to support diskless clients. Requires the package `ltsp-server`.

```
Host ~ $ sudo apt-get install ltsp-server
```

This package contains tools to make a LTSP environment (<http://wiki.ubuntuusers.de/LTSP> install). The following command installs an LTSP environment in `/opt/ltsp/i386`.

```
Host ~ $ sudo ltsp-build-client
```

By Network Block Device (NBD), the LTSP image is provided. The command `sudo ltsp-update-image` generates this in the `/opt/ltsp/images`. It requires that the services are restarted.

```
Host ~ $ sudo ltsp-update-image - arch i386
Host ~ $ sudo /etc/init.d/dhcdbd restart
Host ~ $ sudo /etc/init.d/tftpd-hpa restart
Host ~ $ sudo /etc/init.d/openbsd-inetd restart
```

QEMU is started with the following options.

```
Host ~ $ sudo qemu-net nic-net tap-n-vga std boot-monitor stdio
```

VNC

Virtual Network Computing (VNC) allows the screen of a computer (on which the VNC server running) to another computer (on which the VNC viewer is running) show in exchange for keyboard and mouse movements to send back. In order for a remote access is possible on the desktop of another computer. Since there are VNC server and viewer for nearly all major operating systems, remote access to computers with other operating systems are possible. Another advantage of VNC is that programs started continue to run independently of the VNC session. When re-logging in via VNC you can find the desktop back in front just as you left it. QEMU and KVM contain an internal VNC server of a remote access to the host system allows. The guest system does not need to VNC server installed. To connect to the VNC server VNC Viewer is required. For the Linux distribution Ubuntu desktop is already installed as such. He is called the *Applications menu, Internet, Terminal Server Client*. If there is no VNC Viewer is available on the system, (is TightVNC <http://www.tightvnc.com>) installed. The call on Microsoft Windows via the *Start menu, Programs, TightVNC, TightVNC Viewer (Fast Compression)*. For example, my installation on a Linux Debian derivative.

```
Host ~ $ sudo apt-get install xtightvncviewer
```

The VNC server activates *the-vnc* followed by the host name and a number. The number indicates the VNC server on the host computer, as several VNC servers can be accessed simultaneously. *The-k us* on the German keyboard.

```
Host ~ $ qemu Platte.img-k de-vnc localhost: 1
```

In the *option-vnc* seems no video. It starts a VNC viewer. Running VNC server and VNC viewer on the same machine, there is an option for the VNC Viewer *localhost: a 1*. Is the VNC server on a different computer, the computer name or IP address to be indicated. You can also start a second instance of an enabled VNC server. To do this, after *the-vnc* entered a *second*

```
Host ~ $ qemu Platte.img-k de-vnc localhost: 2
```

The *option-sdl* SDL is also enabled with the graphic output.

```
Host ~ $ qemu Platte.img-k de-vnc localhost: 1-sdl
```

Useful in conjunction with *the-vnc option USBDevice the tablet*. This provides a special mouse emulation, the mouse pointer no longer holds in the window of the instance. If you quit the window with the mouse pointer to control the mouse pointer is again taken over by the host system.

```
Host ~ $ qemu Platte.img-k de-vnc: 1-USBDevice tablet
```

The *lossy* option (from QEMU 0.13.0) allows a lossy compression method (gradient, JPEG, ...). If this option is enabled, there may be lossy updates the screen layout. It will be taking a much lower bandwidth.

```
Host ~ $ qemu Platte.img-k de-vnc: 1, lossy
```

In the QEMU monitor, the command *info vnc* information from status.

```
(Qemu) info vnc
VNC server active on: 0.0.0.0:1
No client connected
```

The activation of the VNC server can *option-vnc none* are prevented with the. The configuration of the VNC server is then in QEMU manager with the command *change vnc*. The parameters are the same as *option-vnc*.

```
Host ~ $ qemu-k en-vnc Platte.img none-monitor stdio
(Qemu) info vnc
VNC server disabled
(Qemu) change vnc localhost: 1
```

Also, a reverse VNC connection is possible. To this end a VNC viewer in listen mode is to start. An instance can then select *reverse* with the VNC viewer to connect the. This example is run on a Windows PC and a VNC client TightVNC in listen mode. Then, an instance on a Linux machine, with the option of *reverse* start here. This is the host name, *Windows PC*, and the port to be declared here. The default value for the port is 5500th If everything

is configured correctly, the VNC issue will appear on the target computer.

```
Host ~ $ qemu-vnc Platte.img windows-pc: 5500, reverse
```

A VNC connection should be secured. A simple way is to restrict access with a password. The VNC protocol limits the length of the passwords to eight characters, and offers little protection. The password protection is enabled with the *password* parameter with.

```
Host ~ $ qemu-monitor stdio Platte.img-vnc: 1, password
```

The password is QEMU monitor with the command *change vnc password* set in the.

```
(Qemu) change vnc password Password: *****
```

A further protection, the authentication with x509 certificates and encryption of the session with TLS (Transport Layer Security). These parameters are the *x509* and *tls*. The required certificates must be in the specified directory, here // *pki / qemu*, etc are located.

```
Host ~ $ qemu-hda-hd-monitor stdio image.img \
-Vnc: 1, tls, x509 = / etc / pki / qemu
```

The parameter *x509verify* server asks the client in addition to the valid certificate.

```
Host ~ $ qemu-hda-hd-monitor stdio image.img \
-Vnc: 1, tls, x509verify = / etc / pki / qemu
```

The highest security is achieved by the additional definition of a password.

```
Host ~ $ qemu-hda-hd-monitor stdio image.img \
-Vnc: 1, password, tls x509verify = / etc / pki / qemu
```

To generate the certificate program is the *certtool*. The installation on Ubuntu is done with a command line.

```
Host ~ # apt-get install gnutls-bin
```

Alternatively, to load from the site *certtool* http://www.gnu.org/software/gnutls/ download and install it. The Windows version of the URL http://josefsson.org/gnutls4win/ to belong. These instructions are for the Linux version. It is recommended for the certificates to create a separate directory. These can be either user *root* in the / *etc / pki / qemu* or other user in \$ *HOME / .pki / qemu* are stored.

```
Host ~ # mkdir-p / etc / pki / qemu
Host ~ # cd / etc / pki / qemu
```

For signing the Keys is a Certificate Authority (CA) is necessary. This may be an official or a self-signed CA. In this example, a self-signed CA is generated. This unique private key for the CA must be created. This key must be kept secure. No way he can get into the wrong hands.

```
Host ~ # certtool - generate-privkey > ca-key.pem
```

This key is set to read only for the owner.

```
Host ~ # chmod 0600 ca-key.pem
```

Based on the private key and configuration information for the organization and a public key is created. The information is written to a text file. The name of the file is *ca.info*. Here is an example for their content.

```
cn = name of their organization
ca
cert_signing_key
```

It is *certtool* the public key created with.

```
Host ~ # certtool - generate-self-signed \
- Load-ca-key.pem privkey \
- Template ca.info \
- Outfile ca-cert.pem
```

Public Key (*ca-cert.pem*) must be on all servers and VNC clients are copied. The hosts with installed QEMU or KVM are referred to as a server. Furthermore, each server needs a private key and certificate. The private key is generated with the command.

```
Host ~ # certtool - generate-privkey > server-key.pem
```

This private key is only port on the designated server. There he is set to read only for the owner.

```
Host ~ # chmod 0600 server-key.pem
```

Also, each server needs a certificate. The necessary information is written to a text file. The name of the file is *server.info*. Here is an example for their content.

```
organization = name of their organization
cn = server.foo.example.com
tls_www_server
encryption_key
signing_key
```

It is *certtool* the certificate generated.

```
Host ~ # certtool - generate-certificate \
          - Load-ca-certificate ca-cert.pem \
          - Load-ca-ca-key.pem privkey \
          - Load-privkey server-key.pem \
          - Template server.info \
          - Outfile server-cert.pem
```

The certificate *server-cert.pem* is provided on the server just created. It must still receive the VNC client your keys and certificates. The private key is created with this command.

```
Host ~ # certtool - generate-privkey > client-key.pem
```

This key is only created on the intended client. There he is set to read only for the owner.

```
Host ~ # chmod 0600-client-key.pem
```

If the option is *x509verify* used in the server, the client needs a certificate. The necessary information is written to a text file. The name of the file is *client.info*. Here is an example for their content.

```
country = DE
state = Berlin
locality = Berlin
organization = name of their organization
cn = client.foo.example.com
tls_www_client
encryption_key
signing_key
```

It is *certtool* the certificate created with and on the VNC client provided copies only.

```
Host ~ # certtool - generate-certificate \
          - Load-ca-certificate ca-cert.pem \
          - Load-ca-ca-key.pem privkey \
          - Load-privkey client-key.pem \
          - Template client.info \
          - Outfile client-cert.pem
```

QEMU and KVM support the authentication of the VNC access with the Simple Authentication and Security Layer (SASL). SASL provides the application protocol is a standard way of negotiating communication parameters between the server and client. It can be used including authentication methods, PAM, GSSAPI / Kerberos, LDAP, SQL database and one-time keys. By configuring the SASL implementations, a method is defined. Was QEMU / KVM support for SASL compiled with this is done by the SASL configuration file / *etc/sasl2/qemu.conf*. Alternatively, another configuration file with the variable *SASL_CONF_PATH* be referred to one. For a simple encryption of passwords, the file / *etc/sasl2/qemu.conf* following content.

```
# / Etc/sasl2/qemu.conf
mech_list: digest-md5
sasldb_path: / etc/sasl2/qemu.db
```

To define SASL passwords is the package *sasl2-bin* install (eg Ubuntu).

```
Host ~ $ sudo apt-get install sasl2-bin
```

The program *saslpasswd2* database, the respective access to the SASL-written. Here, the user *I* created the password *secret* with.

```
Host ~ $ sudo saslpasswd2-f / I etc/sasl2/qemu.db
Password: secret
Again (for verification): secret
```

By the way, do you delete a user in the SASL database with *the-d* option.

```
Host ~ $ saslpasswd2-f / etc/sasl2/qemu.db-I d
```

The scale displayed on one list with the following command.

```
Host ~ $ saslpasswd2-f / etc/sasl2/qemu.db
ich@localhost.localdomain: user password
```

QEMU or KVM is with *the-vnc: 1*, start *sasl*.

```
Host ~ $ qemu-drive-vnc: 1, sasl
```

Unfortunately, not all VNC clients dominate the SASL protocol. In addition to the *virt-viewer* and the *virt-manager* (see Section *libvirt*) version *vinagre* 2.26.1 dominated the SASL protocol.

```
Host ~ $ sudo apt-get install vinagre
Host ~ $ vinagre
```

Additional security from the use of ACLs (Access Control Lists - ACL). have been given QEMU / KVM also be compiled with support for ACL. In x509 client certificates is testing the Distinguished Name. This, for example, the form *C = DE, O = BLA, L = London, CN = I*. When the SASL user name is checked. Depending on the type SASL plugin user can, for example, in the forms *I ich@beispiel.de* be specified, or. If the *acl* flag is set, the access control list is empty and only has the Deny policy. This will prevent anyone from unauthorized access is replaced as long as the VNC server has not loaded any ACL. Enabled option is to use ACL with the *ACL*.

```
Host ~ $ qemu-drive-vnc: 1, sasl, acl
```

To view the access control list *acl_show* used commands in QEMU monitor. This lists all the rules of the ACL and the default policy. There are two named access control lists: the Distinguished Name tests reviewed during *vnc.x509dname vnc.username* to the SASL user name.

```
(Qemu) acl_show vnc.username
policy: deny
```

With the following command will list the *vnc.username* access is granted.

```
(Qemu) acl_policy vnc.username allow
acl: policy set to 'allow'

(Qemu) acl_show vnc.username
policy: allow
```

I The user has access permission.

```
(Qemu) I acl_add vnc.username allow
acl: added a rule at position
(Qemu) acl_show vnc.username
policy: allow
1: I allow
```

The user can *I* be a VNC client with SASL support of the instance to connect to. The access we closed with the

following command.

```
(Qemu) I acl_remove vnc.username  
acl: removed rule at position 1
```

With the following command will list the *vnc.username* access rights revoked.

```
(Qemu) acl_reset vnc.username acl: removed all rules (qemu) acl_show vnc.username policy: deny  
<<< | # # # | >>> http://www.qemu-buch.de
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Netzwerkoptionen/_Netzwerkdienste "

This page has been accessed 14,213 times. This page was last updated on 18 January 2011 at 16:43 clock changed. Content is available under GNU Free Documentation License 1.2 .

Network Protocol Analysis IT forensics computer forensics honeypots QEMU

Kernel-based Virtual Machine tcpdump wireshark libpcap

(Link to this page as [[QEMU-KVM-Buch / Network Options / Network protocol analysis]])

<<< | # # # | >>> | English

Network protocol analysis

Network protocol analysis is needed, among other in the following cases:

- In the IT Forensics the communications of the seized computers to determine facts are examined.
- With honeypots, which are not protected at the network related (virtual) computer criminals are lured.
- Programs that communicate over the network can be tested for errors.
- Undocumented protocols and "phone home" undesirable can be analyzed.

QEMU and the Kernel-based Virtual Machine support the network-protocol analysis by storing the network traffic. The file format is used *libpcap*. This data can be analyzed with the tools *tcpdump* or *Wireshark*. To an instance of a dump in a file to save the network traffic, is *the-net dump*.

```
Host ~ $ qemu-net Platte.img dump
```

In the QEMU monitor console with *info*, this configuration appears *network*.

```
(Qemu) info network
VLAN 0 devices:
user.0: net = 10.0.2.0, restricted = n
dump.0: dump to qemu-vlan0.pcap (len = 65536)
ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

It will dump the network traffic in the file *qemu-wrote the vlan0.pcap*. This is the default setting. QEMU monitor can be set with the command *host_net_remove* the dump be shut down.

```
(Qemu) host_net_remove 0 dump.0
(Qemu) info network
VLAN 0 devices:
user.0: net = 10.0.2.0, restricted = n
ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:12:34:56
```

The command will dump the *dump host_net_add* enabled.

```
(Qemu) host_net_add dump (qemu) info network VLAN 0 devices: user.0: net = 10.0.2.0, restricted = n ne2k_pci.0: model = ne2k_pci, macaddr = 52:54:00:1:
```

The complete syntax for *dump-net* is:

```
Net-dump [, vlan = n] [file = f] [, len = n]
```

Here, the data of the network traffic from VLAN *s* in the *f* file (Default = *qemu-vlan0.pcap*) written. It can with *n* the maximum number of bytes per packet is set (default = 64k).

Analysis with Wireshark

To evaluate the dump data, the tool Wireshark (<http://www.wireshark.org/download.html>) are used. The installation is done in Debian / Ubuntu using a command line.

```
Host ~ $ sudo apt-get install wireshark
```

It calls on Wireshark.

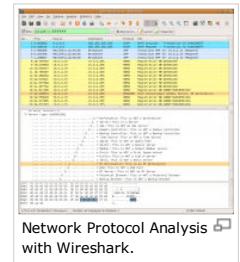
```
Host ~ # wireshark
```

The QEMU / KVM *qemu-written* file will *vlan0.pcap* from the *File menu*, *Open* round. *Reload this icon* on the *capture file* to recent entries in the *file-vlan0.pcap* displayed *qemu*.

<<< | # # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Netzwerkoptionen/_Netzwerk-Protokoll-Analyse"

This page has been accessed 3532 times. This page was last updated on 9 December 2010 at 10:46 clock changed. Content is available under GNU Free Documentation License 1.2 .



Bluetooth USB adapter driver software, Host Controller Interface HCI, BlueZ L2CAP SDP hciconfig hcitool FHSS

(Link to this page as [[QEMU-KVM-Buch / Network Options / Bluetooth]])

<<< | # # # | >>> | English

Bluetooth

The industry standard is the Bluetooth wireless networking devices (cell phones, PDAs, computers and peripherals) for short distance, where both data and voice transfer. The main goal is to replace cable connections between devices. Bluetooth requires special hardware that is built on some computers already. Bluetooth hardware is attached in the form of USB flash drives or PCMCIA cards. Every Bluetooth device has a unique address (Bluetooth Device Address), comparable to the MAC address of a network component. Bluetooth uses a transmission method, the FHSS (Frequency Hopping Fast. This is 1600 times per second, changed the frequency. This makes Bluetooth less sensitive to interference and unauthorized access.

Networks that connect only over a short distance, called the Personal Area Network (PAN). When the Bluetooth piconet is the simplest form of communication link. A Bluetooth device takes the role of the master and the other the role as a slave. The role allocation is random and can be changed during a connection. In a piconet, up to seven active Bluetooth devices are included as a slave. Further still, up to 247 passive slaves exist in a piconet. These slaves listen only to the data of the master. Communication takes place via the master. Slaves can not communicate directly with each other. For several piconets, a scatternet can be formed. It communicate with Bluetooth devices of different piconets on their respective master. While one may have only one piconet master, a slave can participate in multiple piconets. It is possible that a Bluetooth device works in a piconet as a slave and the other piconet as master.

Bluetooth provides for different applications, different transmission paths. This allows a protocol stack that offers a variety of different layers of protocols. The lower layers provide access to higher layers on the radio medium. The Host Controller Interface (HCI) is the command interface for higher layers to access the lower layers. The Logical Link Control and Adaptation Protocol (L2CAP) provides several logical channels are available and segmented great news for the transport. The Service Discovery Protocol (SDP) allows the identification of services of other Bluetooth devices. The transmission of audio signals directly by the lower classes are available.

Software packages with drivers to connect to Bluetooth called Bluetooth stack. Starting with Microsoft Windows XP SP2, it is no longer required a specific Bluetooth driver installed. Opportunities are also using the Bluetooth stack of others. Other operating systems (Linux, Mac OS X) have their own Bluetooth stack.

BlueZ

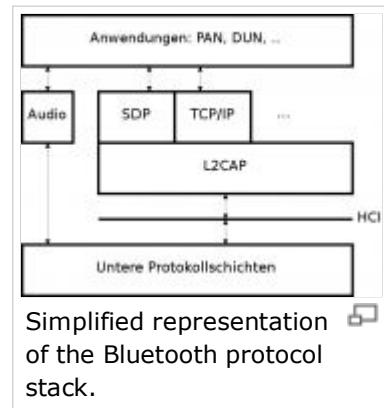
Website: <http://www.bluez.org>

In current Linux distributions is the standard BlueZ Bluetooth stack. In many Linux distributions BlueZ is installed. It is recommended to install these packages (eg Ubuntu).

```
Host ~ $ sudo apt-get install bluetooth bluez bluez-utils \
bluez-hcidump btscanner grml-btnet libbluetooth3 libbluetooth-dev
```

First, check that, if the Bluetooth daemon is active.

```
Host ~ $ ps aux | grep Bluetooth [d]
root 8010 0.0 0.0 3488 1680? Ss 20:02 0:00 /usr/sbin/bluetoothd
```



By connecting to a Bluetooth device, you check with the command `hciconfig hci0` the newly created interface.

```
~ $ Host-a hciconfig
hci0: Type: USB ...
```

If the Bluetooth device is not detected, *messages* to help the entries in the log file / var / log / for debugging. For example, the following messages appear when inserting a Bluetooth USB stick.

```
Host ~ $ tail-f / var / log / messages
... usb 4-1: new full speed USB device using uhci_hcd and address 3
... usb 4-1: configuration # 1 chosen from 1 choice
```

The names of the physical HCI are the command `hcitool dev` identified with.

```
Host ~ $ hcitool dev
Devices: hci0 XX: XX: XX: XX: XX: XX
```

It seems the Bluetooth device with its Bluetooth Device Address. The compound is tested with *I2ping*.

```
Host ~ $ i2ping XX: XX: XX: XX: XX: XX
```

With `hcidump btscanner` or monitored to the HCI packets of a connected Bluetooth devices.

```
Host ~ $ sudo hcidump-x
```

With the command `hcitool scan` to determine whether available Bluetooth devices in range.

```
Host ~ $ hcitool scan
Scanning ...
XX: XX: XX: XX: XX: XX Bluetooth device
```

The-bt

0.10.0 started with QEMU support Bluetooth emulation. *The-bt hci* defines the functions of the corresponding Bluetooth Host Controller Interface (HCI). The parameters of the HCI *bt* be assigned in the host system. If in a virtual machine is just a example to emulate HCI is only the first definition given by *bt hci [...] valid*. The transport layer is determined by the machine type. Currently available only from the Nokia N800 and N810 emulated Internet Tablets (*qemu-system-arm-M n800*, *qemu-system-arm-M n810*) an HCI. In the default *setting-bt hci, null*, the corresponding host Bluetooth HCI no internal logic and thus not responding to commands or events.

```
Host ~ $ qemu-Platte.img bt hci, null
```

If the *option-bt hci, host [: id]* events forwarded to the corresponding host Bluetooth HCI commands and to and from the physical HCI named *id* (default = hci0) the host system to continue. The option is only available on systems with *BlueZ* (Linux Bluetooth protocol stack) is available. From the instance to the device *hci0* be addressed in this case.

```
Host ~ $ qemu-system-arm-M-Platte.img bt hci, host-n800: hci0
```

Since *hci0* the default device is, this information may be omitted.

```
Host ~ $ qemu-system-arm-M-n800 Platte.img bt hci, host
```

With *the-bt hci [, vlan = n]* a, standard HCI created virtual, the Bluetooth scatternet *n* (default = 0) takes part in the.

```
Host ~ $ qemu-Platte.img bt hci, vlan = 0
```

The-bt vhci [, vlan = n] is only available on Linux, and places in HCI Bluetooth Scatternet *n* (default = 0). The HCI is connected to the Bluetooth stack of the host system. This communication between host and guest systems enables a common scatternet. This must be installed the Linux driver VHCI (Virtual Host

Controller Interconnect). Example:

```
Host ~ $ qemu-Platte.img bt hci, vlan = 5-bt vhci, vlan = 5
```

With *the-bt device: dev [, vlan = n]* is a Bluetooth device on the network and emulate *dev n* (default = 0) placed. QEMU supports only the device *keyboard*. It emulates a virtual wireless keyboard with the Human Interface Device Protocol (HIDP).

```
Host ~ $ qemu-Platte.img bt device: keyboard
```

The *option-USBDevice bt [: hci-type]* dongle is a USB Bluetooth emulated. The parameters correspond to those *of-bt hci*. If *hci* type not specified, the default *setting-bt hci, vlan = 0*. This USB Device USB implements the Transport Layer of HCI. Example:

```
Host ~ $ qemu-Platte.img USBDevice bt: hci, vlan = 3 \
Bt-device: keyboard, vlan = 3
```

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Netzwerkoptionen/_Bluetooth "

This page has been accessed 5812 times. This page was last updated on 27 September 2010 at 06:18 clock changed. Content is available under GNU Free Documentation License 1.2 .

Debug and expert options, live migration, kernel debugger GDB software breakpoint, logging, vmlinuz initrd boot, user-space emulation chroot runas

(Link to this page as [[QEMU-KVM-Buch / Special QEMU options]])

<<< | # # # | >>> | English

Contents	
1 special features	
1.1 Live Migration	
1.1.1 Test on a host system	
Example 1.1.2 Live-CD	
1.1.3 Example NBD-Storage	
1.1.4 Copy of the virtual disks in the migration	
2.1 Debug and expert options	
1.2.1 Redirection of the QEMU monitor	
1.2.2 redirect serial port	
1.2.3 Redirection of the parallel interface	
1.2.4 Logging	
1.2.5 Communication with the GDB debugger	
1.2.6 Pause, Single Step	
1.2.7 Watchdog	
1.2.8 Fault Injection	
1.3 Special Linux boot options	
4.1 User Space Emulation	
1.4.1 On Linux	
1.4.2 Mac OS X / Darwin	
1.4.3 Under BSD Unix	
1.4.4 Analysis of suspected system intrusion	
1.5 Other Options	



Special Options

Live Migration

During the live migration of running virtual machines on a host system are transmitted to another. The host systems are continuously available, even if one of the hosts are shut down for maintenance work. It is recommended that the source and destination hosts have the same architecture. However, it is possible, for example, to migrate between hosts guest systems with AMD and Intel architecture or between Linux and Windows hosts. If the images used for the live migration can not be copied to the source and destination host can access the same image bit-level accuracy. This may be with network file systems, such as to ensure, for example, NFS, iSCSI or NBD. Cheaper access to a highly available Storage Area Network (SAN) with fiber optics. Continue to source and destination host use the same network segment. It is beneficial to use the same paths for each virtual machine on the source and destination host. *The-snapshot* option is not applicable. For the live migration of the following steps:

1. The instance is run on the source host.
2. The destination host is the host instance with the same options as on source start, adding the *option-incoming tcp: ip: port* is indicated. The *ip* parameter specifies the IP address for reception of migration. If a 0 is specified, listening on all interfaces. The *port* parameter defines the port. The instance started waiting in a frozen state.
3. the source host is on the QEMU monitor instance of the command of the *migrate-d tcp: ip: input*

port. The *option-d* migration is not the end of the wait for the prompt again to release. The parameter *ip* is the IP address of the destination host dar. If the migration is successful, the source instance and it can be frozen instance on the target host is working with the.

test on a host system

Live migration can also be tested on a host. This will start an instance in a console.

```
Qemu host $ Platte.img
```

On a second console is a second instance with the additional *option-incoming tcp: 0:1234* called. This waiting for that instance on port 1234 on a migration.

```
Qemu host $ Platte.img-incoming tcp: 0:1234
```

In the first instance you switch to the QEMU monitor, and there are the following command.

```
(Qemu) migrate-d tcp: localhost: 1234
```

After a few seconds, the virtual machine has changed instances. Information on the status of migration, the command *info* to *migrate*.

```
(Qemu) info migrate
```

The command *migrate_cancel* breaks from the current migration.

```
(Qemu) migrate_cancel
```

The command *migrate_set_speed value* limits the maximum speed of migration in bytes.

```
(Qemu) migrate_set_speed 1g
```

The command *migrate_set_downtime second* defines the maximum tolerable downtime for the migration in seconds. In this example, the instance may not be attainable maximum of 2 seconds.

```
(Qemu) migrate_set_downtime 2
```

Example Live CD

The condition that the source and destination host must have access to the same bit-level accuracy image true to in live CD / DVD. This is a common storage is not necessary. In each case, a copy of the live CD / DVD be present on the source and destination host. For certain applications, the use of a custom live CD / DVD is useful as a virtual machine. This started on the source host the live CD / DVD.

```
Source host $ qemu-cdrom live-cd.iso-boot d
```

The destination host is the instance with the additional *option-incoming tcp: 0:1234* called.

```
Target host $ qemu-cdrom live-cd-boot d-incoming tcp: 0:1234
```

The QEMU monitor the source instance is the command to *migrate* the IP address or host name of the destination host and port to use the.

```
(Qemu) migrate-d tcp: host-target: 1234
```

Example NBD-Storage

The tool *qemu-nbd* can be an image as a network block device on the network available to make. On the machine where the image is to start *qemu-nbd*. The option *-shared = num* allows multiple instances of network block device can access an on. The maximum number defines the value of *num* (Default = 1).

```
Storage host ~ $ qemu-nbd-t-p 1025 - share = 2 Platte.img
```

On the source host, the instance is started by accessing the Network Block Device.

```
Source host ~ $ qemu-hda nbd: storage host: 1025
```

The destination host is the start of the instance in *addition*, the *option-incoming*.

```
Target host ~ $ qemu-hda nbd: storage host: 1025-incoming tcp: 0:1234
```

The QEMU monitor the source instance is the command to *migrate* the IP address or host name of the destination host and port to use the.

```
(Qemu) migrate-d tcp: host-target: 1234
```

Copy of the virtual disks in the migration

The *-b* option enables the copying of the virtual disks to the destination host. This is useful if the image is not on a common storage (shared storage). The *option-i* is only a copy of the overlay files. This is useful if the same base images are the source and destination hosts. The destination host is a blank image with at least the same capacity as the image on the source host is created.

```
Target host ~ $ qemu-img create Platte.img 1G
```

QEMU, with this blank image and the *option-start incoming*.

```
Target host ~ $ qemu-hda Platte.img-incoming tcp: 0:1234
```

On the source host to QEMU starts to copy the image.

```
Source host ~ $ qemu Platte.img
```

The QEMU monitor the source instance to start the live migration and copying of the image.

```
(Qemu) migrate-b tcp: host-target: 1234
```

Debug and expert options

redirect the QEMU monitor

In the default setting of the QEMU monitor on the second console of the instance is run. The key combination [Ctrl] + [Alt] + [2] switches on this console. This default setting is the startup *option-monitor vc*.

```
Host ~ $ qemu-monitor Platte.img vc
```

The QEMU monitor, the *option-monitor* and the subsequent indication of a target devices redirected to. He will then no longer on the second console. In this example, the Linux QEMU monitor *stdio* redirected to (*stdio* is the standard input and output of the host system). That is, the QEMU monitor is redirected to the console of the host system.

```
Host ~ $ qemu-monitor stdio Platte.img  
(Qemu)
```

It is possible the QEMU monitor on a port to redirect the host system. In this example, the QEMU monitor on port 4444 of the host system as the Telnet protocol will be offered. The *nowait* parameter causes that instance without a Telnet connection also starts.

```
Host ~ $ qemu-monitor Platte.img tcp:: 4444, server nowait
```

With a Telnet client connects you to the port 4444 of the host system.

```
Host ~ $ telnet localhost 4444  
Escape character is'^]'.  
(Qemu)
```

redirect the serial port

In the default setting, the virtual serial port on the third console is run. The key combination [Ctrl] + [Alt] + [3] switches on this console. The default setting corresponds to the QEMU start *option-serial vc*.

```
Host ~ $ qemu-serial vc Platte.img
```

The output of the virtual serial port, the *option-serial* and the subsequent indication of a target devices redirected to. You will then no longer on the third console. In this example, is redirected to the serial port of the Linux host system to the first serial port on the host system.

```
Host ~ $ qemu-Platte.img serial / dev/ttys0
```

On Microsoft Windows as the host system interface is the first serial *COM1* designated.

```
Host C: \> Platte.img qemu-serial COM1
```

To redirect to a file, use the parameter *file:* followed by a file name.

```
Host ~ $ qemu-serial Platte.img file: Seriell.log
```

Disabled interface is the virtual *serial-serial none* with.

```
Host ~ $ qemu-serial none Platte.img
```

The-serial can be specified once in four to. Other configurations of the serial interface are listed in the Annex.

Redirect parallel port

In the default setting, the virtual parallel port is set to the fourth console. The key combination [Ctrl] + [Alt] + [4] switches on this console. The default setting corresponds to the QEMU start *option-parallel vc*.

```
Host ~ $ qemu-parallel Platte.img vc
```

The output of the virtual parallel port is the *option-parallel* and the subsequent indication of a target devices redirected to. You will then no longer on the fourth console. In this example, will be diverted to the parallel port of the Linux host system to the second serial port on the host system.

```
Host ~ $ qemu-Platte.img parallel / dev/ttys1
```

On Microsoft Windows *COM2*, the second serial interface means.

```
Host C: \> qemu Platte.img parallel-COM2
```

To redirect to a file, use the parameter *file:* followed by a file name.

```
Host ~ $ qemu-Platte.img parallel file: Parallel.log
```

On Linux host system, the parallel ports / *dev/parport0* up / *dev/parport2* be addressed directly. You need to account under QEMU its rights or the Kernel-based Virtual Machine is running, read and write permissions are assigned. Here is the example of the device / *dev/parport0*:

```
Host ~ $ sudo chmod o + rw / dev/parport0
```

QEMU the Kernel-based Virtual Machine or the *option-parallel / dev/parport0* started.

```
Host ~ $ qemu-rtc Platte.img base = local time-parallel / dev/parport0
```

In the host system so that the parallel port, a host of connected printers are used. Interface is disabled, the virtual *parallel-parallel* with *none*.

```
Host ~ $ qemu Platte.img-parallel none
```

The *parallel* option can be given once in three to.

Logging

The *-d* option enables the writing of log data in the file / tmp / *qemu.log*. Possible log entries shows the *-d option*?

```
Host ~ $ qemu-d?
Log items (comma separated):
out_asm show host generated assembly code for each compiled TB
in_asm show target assembly code for each compiled TB
op show micro ops for each compiled TB
op_opt show micro ops
int show interrupts / exceptions in short format
exec show trace each executed before TB (lots of logs)
show cpu CPU state before block translation
pcall show protected mode far calls / returns / exceptions
cpu_reset show CPU CPU state before resets
```

There are multiple entries possible.

```
Host ~ $ qemu Platte.img-d, int cpu
```

With the command *tail* can view the log data.

```
Host ~ $ tail-f / tmp / qemu.log
```

Communication with the GDB debugger

The most common debugger in the open source world is the GDB (GNU debugger). With it, programs written with different programming languages test. The call is made with the program to be tested and further arguments. GDB is able to remotely access programs and processes. In addition to the historically used for serial interfaces like this can be done via a TCP / IP connection. This is on the remote system to start the program *gdbserver*. GDB you call as usual on the host computer.

```
Host ~ $ gdb
(Gdb)
```

In GDB then the connection to the remote system. The default port for remote debugging is 1234

```
(Gdb) target remote <remotehost>: 1234
Remote debugging using: 1234
0x901a7a4d in? ()
```

Would like to use the GDB, the KVM is necessary to *gdbserver* in QEMU or integrated. It is activated with the *-s* option.

```
Host ~ $ qemu-s Platte.img
```

If the *option-s* not specified, *gdbserver* is activated in the QEMU monitor.

```
(Qemu) gdbserver
```

Optionally, a different port to be defined.

```
(Qemu) gdbserver 4321
```

Gdbserver was activated, the guest system is stopped and the control is carried out by the debugger. The command *c* (*continue*) wakes up the host system.

It can be a QEMU or KVM instance in the debugger GDB control and start. With the *option-gdb dev* instance expects a GDB connection on the device to *dev*. Typical compounds are based on TCP. Possible but also UDP, TCP and pseudo-stdio. In the following example, a QEMU instance is started in the GDB and

controlled.

```
Host ~ $ gdb
(Gdb) target remote | exec qemu-gdb Platte.img stdio
Remote debugging using | exec qemu-gdb Platte.img stdio
[New Thread 1]
0x0000ffff0 in?  ()
(Gdb) help
...
(Gdb) c
Continuing.
^ c
Program received signal SIGINT, Interrupt.
0x800c7058 in?  ()
(Gdb) stop
(Gdb) quit
The program is running.  Exit anyway? (Y or n) y
```

Systems that are compiled without debugging information, can be most difficult. There is no information about the program status is available. One sees the contents of processor registers, without information about the program benefits not that much. Given, however, specially prepared systems for debugging source code together, you have a powerful tool at hand.

Pause, Single Step

The-S freezes the CPU (s) Start at the instance of a.

```
Host ~ $ qemu-S Platte.img
```

The QEMU monitor *status* information about the status of the command *info*.

```
(Qemu) info status
VM status: paused
```

Aroused the instance with the commands *c* or *cont*.

```
(Qemu) cont
(Qemu) info status
VM status: running
```

The *stop* command can pause the instance again.

```
(Qemu) stop
```

With the *option-single step* mode, the instance for debugging in single-step start.

```
Host ~ $ qemu Platte.img-single step
```

The QEMU monitor informs the command *info status* if the single step mode is enabled or not.

```
(Qemu) info status
VM status: running (single step mode)
```

In the QEMU monitor, the command *single step [on | off]*, the single step mode during the term of the instance (de-) activated. In this example, the single step mode is disabled.

```
(Qemu) single step off
```

Watchdog

A Watchdog (watchdog) is used to monitor a system. When detected malfunctions a certain action is triggered. Usually a hardware watchdog is used. If the host system, an agent for the watchdog is

activated, the agent must make periodic requests to the watchdog. If these questions from, so will the watchdog of a given action. With the *option-watchdog* watchdog *model* is a virtual hardware-enabled. The parameter *model* selects the model to be emulated. The models available lists *on-watchdog?*. The choice depends on the support of the model by the host system.

```
Host ~ $ qemu-watchdog?
i6300esb Intel 6300ESB
ib700 iBASE 700
```

In the following example, the watchdog hardware iBASE 700 is emulated.

```
Host ~ $ qemu disk-watchdog ib700
```

If the watchdog timer alert, is raised by default to reset. With the *option-watchdog-action* suits to the action on. Possible actions are *reset* (default), *shutdown* (graceful shutdown), *poweroff*, *pause*, *debug* (output a debug message) or *none* (no action). The following example is the action set *debug*.

```
Host ~ $ qemu disk-watchdog ib700-watchdog-debug action
```

The action is changed in the QEMU monitor with the command *watchdog_action*.

```
(Qemu) watchdog_action none
```

Fault Injection

QEMU supports the simulation of hardware failure. The QEMU monitor the *nmi* command injected a non-maskable interrupt (NMI) for the specified CPU. An NMI is an interrupt, the system can not be ignored. NMI usually work from very fundamental and critical routines.

```
(Qemu) nmi 0
```

The QEMU monitor the injected command) a *mce* Machine Check Exception (MCE. A machine check exception is a report of recent CPU (s) from Intel and AMD hardware problems. For the example we emulate two 64-bit CPU Type AMD Phenom.

```
Host ~ $ qemu-system-x86_64 Platte.img-snapshot \
-Cpu phenom-smp 2-monitor stdio
```

If the host system Linux, so can be tested using the following commands the support of MCE.

```
Guest ~ $ grep -i mce / proc / kallsyms
Guest ~ $ ls -l / dev / mcelog
```

Is the device */dev/mcelog* not exist, if you hold it with *mknod*.

```
Guest ~ $ mknod / dev / mcelog c 10 227
```

The QEMU monitor is an MCE is injected.

```
(Qemu) mce 0 4 0x00 0x00 0x00 0xbe0000001008081f
```

The syntax of *mce* is:

```
(Qemu) mce cpu bank status mcgstatus addr misc
```

The MCE is the specified CPU (*cpu*) for injection. If only one CPU is emulated specify level *0*. Each CPU is four or five banks (*bank*) is assigned. A bank is a group of error registers. This error registers can each be activated or deactivated. Normally all error register of all banks are activated. The number and content of the benches depends on the CPU. Each bank has four registers. In an AMD CPU, the banks have the following meanings:

- Bank 0: Date cache
- Bank 1: Instruction cache
- Bank 2: Bus Unit,

Bank 3: Load Store Unit

Bank 4 North Bridge and DRAM.

With *status*), the bank status register (64-bit is set.

The Global Status Register (*mcgstatus*) contains information whether a MCE was reported.

Bit 0: RIPV - Restart IP valid flag,

Bit 1: EIPV - Error IP valid flag,

Bit 2: MCIP - Machine check in progress,

Bit 3 to 63: Reserved.

With *addr* exception is the memory address of the defined. For additional description of the MCE is *misc*.

Special Linux boot options

With the help of special Linux boot options, a kernel is loaded, without having to be installed on a virtual hard disk. The kernel can be either a Linux kernel, or a multi-boot kernel. This is useful for example to test a compiled kernel. These options for the emulation of other processor architectures are necessary because they use other boot methods.

```
Host ~ $ qemu-kernel vmlinuz-initrd initrd.gz-append "..."
```

The startup option is *kernel-kernel* to boot in the *one-initrd* allows the inclusion of an initial RAM disk. An initial ramdisk (*initrd*) is a temporary filesystem that the kernel during the boot process is used by. It contains the image of a file system mounted with the files needed to boot the system and is used by Linux kernel or other Unix-related operating systems at boot time as root. For embedded systems, all the functionality included in the *initrd* are the system. In a multi-boot *kernel-initrd* kernel modules and their parameters are passed behind. Kernel boot options to define *with-append*. As an example system is used here to boot one of a virtual disk to the script *qemu-make-debian-root* has been created with. This script creates a virtual hard disk and installed the packages for Debian-based system. A kernel is not generated. The script *qemu-make-debian-root* is in Debian distributions such as Ubuntu, the package *qemu*. It requires the packages *debootstrap* and *sharutils*.

```
Host ~ $ sudo apt-get install debootstrap sharutils qemu
```

When calling the script *qemu-make-debian-root* is the size of the virtual disk in MB, the install Debian or Ubuntu version, the download URL and name of the virtual hard disk to be created to. To run this script requires root privileges. To avoid that the image of the user is root, place the virtual disk previously shipped with the *touch* command as normal user.

```
Host ~ $ touch Platte.img
Host ~ $ sudo qemu-make-debian-root 500 sid \
http://ftp.debian.org/debian Platte.img
```

As already described, a kernel available for this virtual disk. To load appropriate kernel and the corresponding initial ramdisk down one, for example, *ftp://ftp.debian.org/debian/dists/sid/main/installer-i386/current/images/hd-media/*. With these options, the instance is started.

```
Host ~ $ qemu-kernel vmlinuz Platte.img \
-Initrd initrd.gz-append "root = / dev / ram"
```

User Space Emulation

In contrast to kernel space that is reserved only for the execution of the kernel and its extensions for drivers that run in user space all user applications. With the user-space emulation software can run in user space, which actually runs in the kernel space. This is useful for the development of such software. If this software is tested directly in kernel space, the stability of the system is compromised. When the user space emulation, which stands for Linux, Mac OS X / Darwin and BSD available, not a full PC is emulated, but it will not support the use of other dynamic libraries. Thus, programs can be started, which were actually compiled libraries of other architectures. This enables developers to test software for other architectures and programming. Since the user space emulation QEMU the started processes is not forecloses, this emulation is not virtualization. In user space can x86, PowerPC, ARM, MIPS 32-bit,

Sparc32/64, ColdFire (m68k), CRISv32 and MicroBlaze CPUs are emulated.

Linux

To better understand the user-space emulation program is useful outputs the information about the system used. On Unix / Linux is used to the command `uname`. If you give command `uname-a` in the command line, the so obtained in this example, the kernel name (*Linux*), the host name (*kissing*), the kernel version (2.6), the creation date of the kernel (# 1 SMP Thu Oct 30 04:57 : 35 UTC 2008), the architecture (*i686*) and the operating system (*GNU / Linux*).

```
Host ~ $ uname -a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 i686 GNU / Linux
```

QEMU package is included in the user space emulator `qemu-i386` for the **x86 architecture**. The *option-L* libraries, the search path to the dynamic defined. Is specified as a search path to the root directory, the native libraries in the host operating system is used. The second option defines the path to the executable. In the following example, the command `uname` of the host system in user space emulator with the native libraries on the host operating system is called.

```
Host ~ $ qemu-i386-L / bin / uname -a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 i686 GNU / Linux
```

Since both the libraries and the binary from the host system is used, the same information from the `uname` here, as if called directly. For illustrative demonstration of the user space emulation you need examples of libraries and binaries for various architectures. These can range from the QEMU website <http://wiki.qemu.org/Download point QEMU Linux user mode emulation tests> on the downloaded. We unpacked this tarball and go into the created directory.

```
Host ~ $ wget http://wiki.qemu.org/download/linux-user-test-0.3.tar.gz
Host ~ $ tar xzvf linux-user-test-0.3.tar.gz
Host $ cd linux-user-test-0.3
```

In the subdirectory *gnemul* are the libraries of different architectures. The binaries of these architectures are in subdirectories. Sun are versions of the command `uname` for multiple architectures.

```
Host ~ $ ls * / uname
alpha / uname arm / hppa uname / uname m68k/uname mipsel / uname
mips / uname ppc / sparc sh4eb/uname uname / uname x86_64/uname
```

The command `file` can you check whether it is really the binaries for other architectures involved. In this example, the command `uname` of the ARM architecture studied.

```
Host ~ $ file arm / uname
arm / uname: ELF 32-bit LSB executable, ARM, version 1 (ARM)
for GNU / Linux 2.2.0, dynamically linked (uses shared libs),
for GNU / Linux 2.2.0, stripped
```

This binary for ARM architecture can not run in an x86 architecture.

```
Host ~ $ arm / uname -a
bash: arm / uname: can not execute binary file
```

To execute these binaries you need the appropriate user-space emulator and libraries for each architecture. Interesting architecture is what information each of the different displays of the command `uname`. For the user space emulation of the ARM architecture, the program requires `qemu-arm`. The *option-L* architecture is the path to the libraries of ARM-configured. Subsequently, the desired sub-directory called binary in the *arm*.

```
Host ~ $ qemu-arm-L gnemul / qemu-arm / arm / uname -a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 unknown UTC 2008 armv5tel
```

Looking at the information that the command `uname` displays, it appears in the following:

- The kernel is identical to the host system.
- The architecture is shown *armv5tel* and not *i686*.
- The operating system is *unknown*) does not recognize (.

It is emulated with the same kernel a different architecture. The user-space emulator *qemu-arm* makes it possible, using the specified libraries, the execution of the command *arm / uname*, although this ARM processor architecture has been compiled for. The user-space emulation is no virtualization, the started processes are not isolated from the host system. For example, it is possible to use the *dd* command of the ARM architecture in userspace emulator, a CD to import.

```
Host ~ $ qemu-arm-L gnemul / qemu-arm / arm / dd if = / dev / cdrom of = cd.iso
```

For the user space emulation of the **x86 architecture** programs are the *qemu-i386* (32-bit) or *qemu-x86_64* (64-bit) is required. They are the respective libraries x86 and x86 binaries indicated.

```
Host ~ $ qemu-i386-L gnemul/qemu-i386 / i386/ls host ~ $ qemu-x86_64-L gnemul/qemu-x86_64
```

For the user-space emulation of the **SPARC architecture** programs are the *qemu-sparc* (32-bit), or *qemu-qemu-sparc32plus-sparc64* (64-bit) is required.

```
Host ~ $ qemu-sparc-L gnemul / qemu-sparc / sparc / uname-a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 sun4 unknown
Host ~ $ qemu-L-sparc32plus gnemul / qemu-sparc / sparc / uname-a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 sun4 unknown
Host ~ $ qemu-sparc64-L gnemul / qemu-sparc / sparc64/ls
```

For the user space emulation of the **ARM architecture** are the programs or *qemu-arm* *qemu armeb-apply*.

```
Host ~ $ qemu-arm-L gnemul / qemu-arm / arm / uname-a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 unknown UTC 2008 armv5tel
Host ~ $ qemu-L-armeb gnemul / qemu-armeb / armeb / ls
```

For the user-space emulation of the **MIPS architecture**, the programs and *qemu-mips* *qemu-mipsel* responsible.

```
Host ~ $ qemu-mips-L gnemul / qemu-mips mips / uname-a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 mips unknown
Host ~ $ qemu-mipsel-L gnemul / qemu-mips / mipsel / uname-a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 mips unknown
```

For the user-space emulation of the **Coldfire processor architecture**, the program requires *qemu-m68k*.

```
Host ~ $ qemu-m68k-L gnemul/qemu-m68k / m68k/uname-a
```

For the user space emulation of the **PPC architecture** programs are the *qemu-ppc*-*qemu-ppc64* *ppc64abi32* or to use *qemu*.

```
Host ~ $ qemu-ppc-L gnemul / qemu-ppc / ppc / uname-a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 ppc unknown
Host ~ $ qemu-L-ppc64abi32 gnemul / qemu-ppc / ppc / uname-a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 ppc unknown
Host ~ $ qemu-ppc64-L gnemul/qemu-ppc64 / ppc64/ls
```

For the user-space emulation of the **Alpha processor architecture** program is the *qemu-alpha* responsible.

```
Host ~ $ qemu-alpha-L gnemul / qemu-alpha / alpha / ls
```

For the user-space emulation of the **SH4 architecture** programs are the *sh4-or-qemu* *qemu sh4eb* apply.

```
Host ~ $ qemu-sh4-L gnemu/qemu-sh4 / sh4/ls
Host ~ $ qemu-L-sh4eb gnemu/qemu-sh4eb / sh4eb/uname-a
Knut Linux 2.6 # 1 SMP Thu Oct 30 04:57:35 UTC 2008 sh4 unknown
```

More User-Space emulators *qemu-cris* (**ETRAX CRIS processor architecture**) and *qemu-MicroBlaze* (**MicroBlaze processor architecture**).

Mac OS X / Darwin

The user space emulation on Mac OS X / Darwin in the same way as under Linux. In addition, the FAT libraries are needed. This may need from the Mac OS X CD to install or compile. The following example starts the PPC version of the binaries:

```
Host ~ $ qemu-ppc / bin / ls
```

In a PPC architecture, QEMU is to specify the path to the x86 libraries.

```
Host ~ $ qemu-i386-L / opt/x86_root / bin / ls
```

Under BSD Unix

Still in development is the user-space emulation for BSD Unix. In QEMU, version 0.10.0 can be used on a Sparc64 host system, the libraries of Sparc64 for binaries.

```
Host ~ $ qemu-sparc64 / bin / ls
```

The user space emulation on BSD Unix also has the *option-type* with the possible parameters *bsd*, *FreeBSD*, *NetBSD* and *OpenBSD* (default). In order to emulate the BSD is set.

Analysis of suspected system intrusion

Besides the advantages for software development is a different application of the user space emulation is interesting. In the case of a break in a system first tries to disguise this. These are both manipulated binaries and libraries of the victim system. That is, for example, the *ls* command shows incorrect information. The system shut down and investigate is not always possible. If only a suspicion before, would lead to shut down a critical server downtime. Also, it could disappear important clues, because the content is lost of memory. Cheaper would be the analysis of system-independent tools.

With the user-space emulation such tools can be started with its libraries from a CD or DVD. To the condition that the system under investigation can not be shut down. The following simplified steps demonstrate how to create such a CD / DVD for Linux. First QEMU is to compile from the sources, the last step (*make install*) is not performed where. Furthermore, the files for the user-space emulation of the QEMU web site to download and unpack. The QEMU binaries and files for the user-space emulation be burned to a CD. On the suspicious computer is mounted the CD and changed the mount point. *The-o exec* allows programs to start from the CD.

```
~ # Mount / dev / cdrom / mnt-o exec
~ # Cd / mnt
```

It starts the user-space emulator from the CD. In this example, under the i386 architecture, the command *ls* its libraries from the CD used. Perhaps, on the hard drive manipulated commands and libraries are not required. The command can not be manipulated, because it is stored on a read-only CD. This enables IT forensics is the system in operation.

```
~ # ./qemu-0.11.0/i386-softmmu/qemu \
L-1/usr/local/gnemu/qemu-i386 i386/ls
```

Other Options

The *option-name* one defines a name for the instance. This name appears in the title bar of the SDL window. In the following example, the instance name *Testuslongus*.

```
Host ~ $ qemu-name Platte.img Testuslongus
```

The QEMU monitor can be called this name.

```
(Qemu) info name
Testuslongus
```

Furthermore, can be specified for the instance under Linux, the process name. This process-name shows, for example, the program at *top*.

```
Host ~ $ qemu-name Platte.img Testuslongus, process = Hamster
Host ~ $ top
...
7257 I 20 0 215m 177m 6252 R 98 8.8 0:39.75 Hamster
...
```

The instance is replaced *with-uuid* UUID (Universally Unique Identifier).

```
Host ~ $ qemu Platte.img-uuid-083e-45a4-571bba42 abc6-f15821718453
```

The QEMU monitor can UUID to query.

```
(Qemu) info uuid
571bba42-083e-45a4-abc6-f15821718453
```

The-pidfile file activates the writing of a PID file. A PID-file contains the Process Identification Number (PID) of the process.

```
Host ~ $ qemu Platte.img-pidfile / tmp / qemu.pid
```

Using the Process Identification Number can uniquely identify the process.

```
Host ~ $ ps a | grep `cat / tmp /` qemu.pid
15 342 pts / 1 R + 1:14 Platte.img qemu-pidfile / tmp / qemu.pid
```

The PID can be, for example, determine whether the process is still running and you can process with the UNIX command *kill* to finish this.

```
Host ~ $ kill -9 `cat / tmp /` qemu.pid
```

A reboot of the host system to prevent *with-no-reboot*. Instead, the instance is terminated.

```
Host ~ $ qemu-no-reboot Platte.img
```

Terminating the instance to shut down the guest system with *the-no-shutdown* prevented.

```
Host ~ $ qemu Platte.img-no-shutdown
```

The *option-daemonize* instance is not as long as the Standardein-/ausgabe separated from, to the devices of QEMU / KVM are able to accept connections. This option is useful for avoiding race conditions during initialization of QEMU / KVM.

```
Host ~ $ qemu Platte.img-daemonize
```

The-runas user is on the QEMU-/KVM-Prozess the specified user. This is useful if, for example an instance in which the *root* user must be started without root privileges should be operated, but then. The *option-you chroot* when you start QEMU / KVM to switch to the specified directory *chroot*.

```
Host ~ # qemu Platte.img-k de-vnc localhost: 1 \
Runas-robert-chroot / VMs / qemu /
```

The *option-writeconfig* file configuration settings of the instance in a text written.

```
Host ~ $ qemu-Platte.img writeconfig Konfiguration.txt
```

The text file in this example has this content:

```
# Qemu config file
[Drive]
"0" index =
media = "disk"
file = "Platte.img"
```

The *option-readconfig* instance, the options from the configuration file starts with.

```
Host ~ $ qemu-readconfig Konfiguration.txt
```

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Spezielle_QEMU-Optionen "

This page has been accessed 14,263 times. This page was last updated on 27 September 2010 at 06:19
clock changed. Content is available under GNU Free Documentation License 1.2 .

KVM QEMU GUI management tools AQEMU libvirt

(Link to this page as [[QEMU-KVM-Buch / Management Tools]])

<<< | # # # | >>> | English

Management Tools

The configuration of virtual machines on the command line is all well and quickly carried out by experts but unfavorable for the occasional user. In order to facilitate the work with QEMU and the Kernel-based Virtual Machine Management tools are being developed. Microsoft Windows is the Qemu Manager for Windows. A description can be found at the URL http://qemu-buch.de/d/QEMU_unter_Microsoft_Windows. QEMU on Mac OS X usually operated with the graphical user interface Q. A description can be found at the URL http://qemu-buch.de/d/QEMU_unter_Mac_OS_X.

AQEMU - A GUI for QEMU and KVM

The library libvirt

 Installation

 The Management Tools

 Virtio modules

 Language Bindings

 oVirt

The library libguestfs

OpenQRM

More Tools for QEMU and KVM

<<< | # # # | >>>

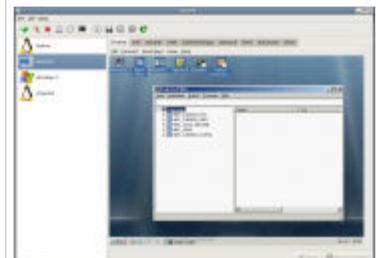
Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools "

This page has been accessed 8899 times. This page was last updated on 27 September 2010 at 06:19
clock changed. Content is available under GNU Free Documentation License 1.2 .

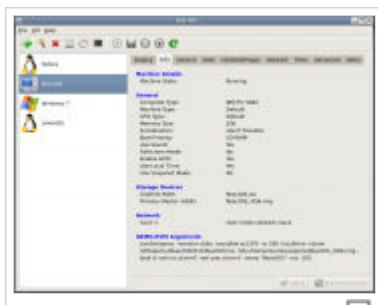
AQEMU 0.8.1 - a graphical frontend (Linux, BSD) and the QEMU for Kernel-based Virtual Machine

(Link to this page as [[QEMU-KVM-Buch / Management Tools / AQEMU]])

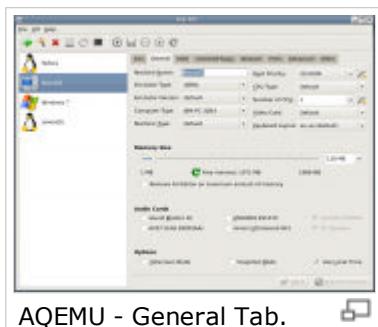
<<< | # # # | >>> | English



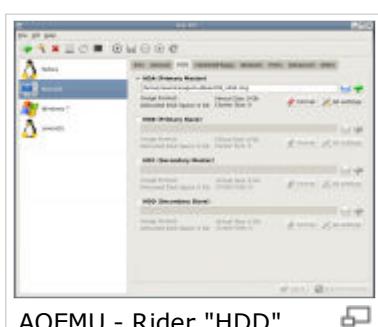
AQEMU - Tab "Display".



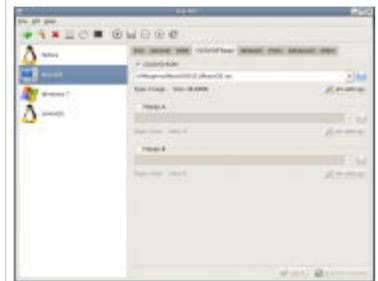
AQEMU - Rider Info.



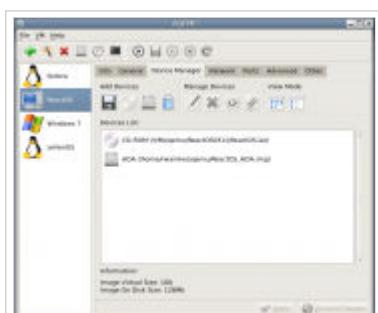
AQEMU - General Tab.



AQEMU - Rider "HDD".



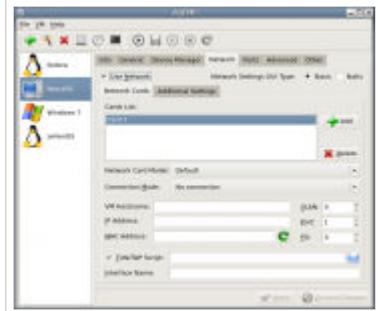
AQEMU - Rider "CD / DVD / Floppy".



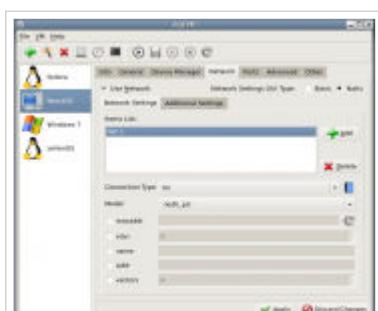
AQEMU - tab "Device Manager".



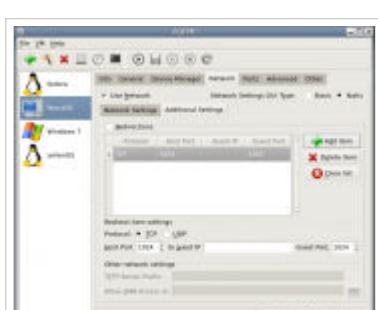
AQEMU - window "New Storage Device".



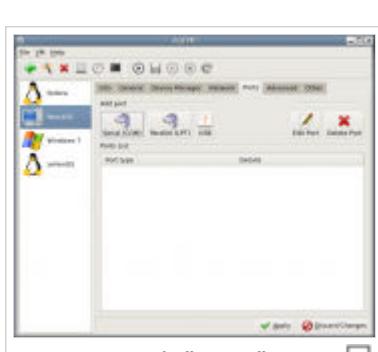
AQEMU - tab "Network" - "cards" - "Basic".



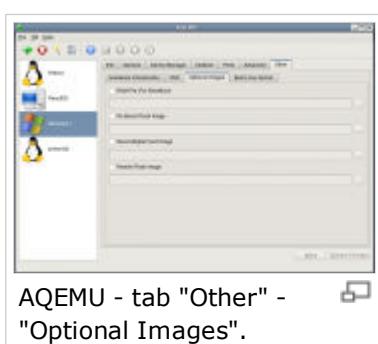
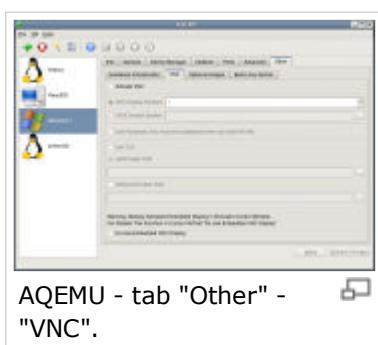
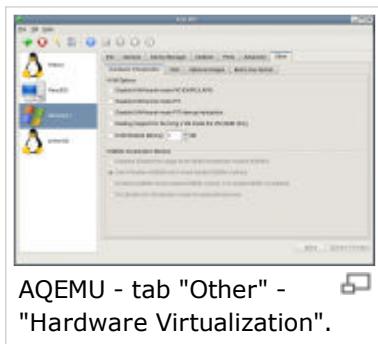
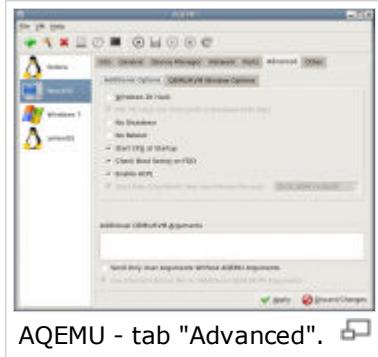
AQEMU - tab "Network" - "cards" - "Native".

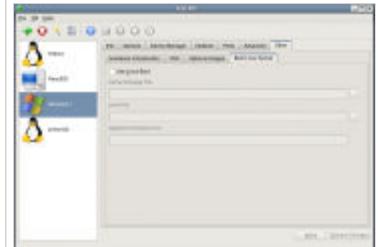


AQEMU - tab "Network" - "Additional Settings".



AQEMU - Tab "Ports".

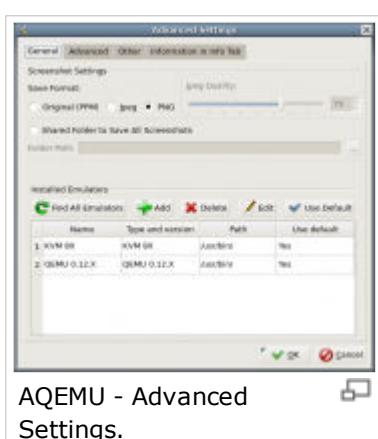




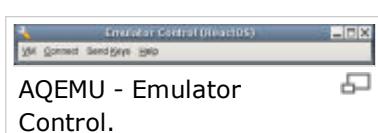
AQEMU - tab "Other" -
"Boot Linux kernel".



AQEMU - Settings.



AQEMU - Advanced
Settings.



AQEMU - Emulator
Control.

Contents

- 1 AQEMU 0.8.1
 - 1.1 The main window
 - 1.2 Windows General Settings
 - 1.3 Advanced Settings window
 - 04/01 Window Create HDD Image
 - Convert Image window 1.5 HDD
 - 1.6 Create Template window
 - 1.7 Window management VM snapshots
 - 1.8 Menus of the VM window (emulator-Control)

AQEMU 0.8.1

Download: <http://sourceforge.net/projects/aqemu/files/>

AQEMU is a powerful front end for QEMU and KVM. AQEMU can be installed on Linux, FreeBSD and PC-BSD. For Linux distributions with the Debian package management (Debian, Ubuntu) is the Debian package download and install it with *dpkg*.

```
Host ~ $ sudo dpkg-i *.deb-aqemu
Host ~ $ sudo apt-get install-f
```

For Gentoo Linux installation is performed with the ebuild.

```
Host ~ # emerge-av app-emulation/aqemu
```

For FreeBSD you are installing with the following commands.

```
Host ~ # cd /usr/ports/emulators/aqemu
Host ~ # make install clean
```

On FreeBSD it is also possible AQEMU to install as a package.

```
Host ~ # pkg_add-r aqemu
```

For other Unix-/Linux-Versionen AQEMU is to compile. This is the Qt library (version 4.2.1) with the extensions needed for the development and the GCC compiler from version 3. For example, my installation on 64-bit Ubuntu.

```
Host ~ $ sudo apt-get install libqt4-dev qt4-dev-tools
        sudo apt-get install g++-dev libvncserver
```

The sources have to be decompressed after downloading and compiling. Then the script is run *install_script.sh*.

```
Host ~ $ tar-* tar.bz2 xjvf aqemu
Host $ cd ~ aqemu *
Host ~ $. / Configure.sh
Host ~ $ make
Host ~ $ sudo make install
Host ~ $ sudo . / Install_script.sh /usr/local/
```

In Gnome AQEMU is the *Applications menu*, *Accessories*, *AQEMU* called on. On the console command is started the GUI with the *aqemu*.

```
Host ~ $ aqemu
```

The first call AQEMU to configure. This is supported by a wizard (*File menu, Run First Start Wizard*). First,

Robert Warnke und Thomas Ritzau

qemu-kvm & libvirt

2010

Warnke, Ritzau

qemu-kvm and libvirt

4. Edition 2010

ISBN: 978-3-8370-0876-0

276 pages, 27.27 EUR

Order



the language must be selected. The next step will be adjusted preferences. The guidelines can be adopted. Then, the paths to the programs QEMU and KVM are determined. This is to click the *search* button. After finishing the Wizards, the main window of AQEMU. New virtual machines I create with a wizard. To do this click on the icon with the wand. For this example (which was the operating system ReactOS <http://www.reactos.org/de/>) selected as the guest. From the ReactOS website is the image file of the installation CD to download and unpack.

```
Host ~ $ unzip ReactOS *-- REL iso.zip
```

The first step is prompted for the configuration mode. As a beginner to using *Typical*. Then choose the type of emulator (QEMU or KVM) from.

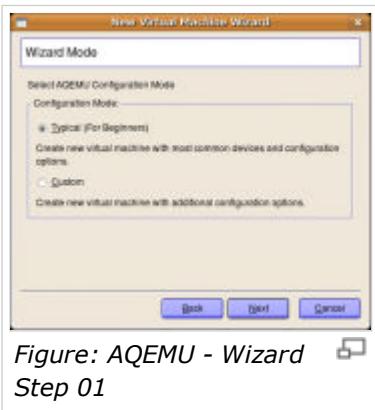


Figure: AQEMU - Wizard

Step 01

The next step you choose a template for the virtual machine. There are several templates for Windows, Linux and BSD versions are available. For this example, the *Windows Template _XP* the right choice.

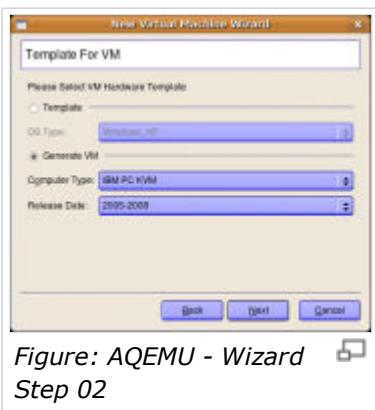


Figure: AQEMU - Wizard

Step 02

The next step in the name of the virtual machine is set: "ReactOS".

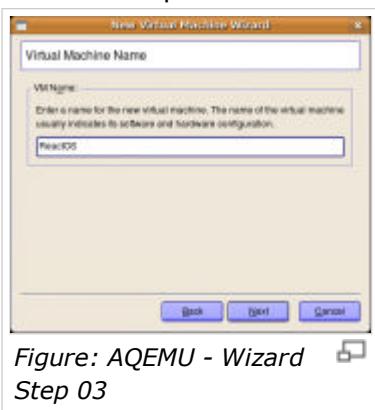


Figure: AQEMU - Wizard

Step 03

Subsequently, the size of the virtual disk is defined. For ReactOS submits GB.



Figure: AQEMU - Wizard

Step 04th

Then the virtual network is set up. For the first test ranges of the *user mode network stack*. To ensure that all steps of the wizard is called. The name of the new virtual machine appears to the left of the list. To install ReactOS is under the *General* tab, the *Boot Priority* to CD-ROM set. The tab *CD / DVD / Floppy Disc* is the path to the downloaded ISO image of installation indicated. With the *Apply* button to save the changes. To start the virtual machine, choose its name in the list and press the start button.

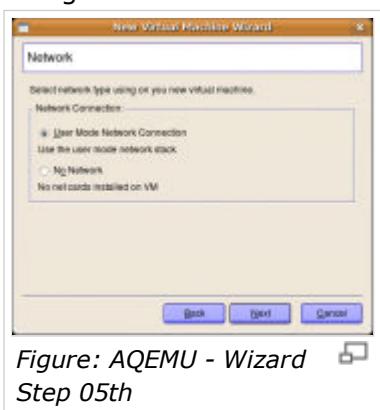


Figure: AQEMU - Wizard

Step 05th

The main window

On the left side of the created virtual machines are listed. To configure a virtual machine, this is to be selected. Subsequently, the desired settings are made under the rider or the context menu (right mouse button on the name). For the most frequently used commands (create, start and stop) are the icons in the toolbar.

The tab *is* displayed only if the selected virtual machine is running and the *File menu*, *General settings*, select *Show in Main AQEMU With Windows Embedded Display* is enabled. The *display* tab allows communication with the host system in the virtual machine. Alternatively, the output will appear in the host system in a separate window. In case of any problems *VNC* is the menu item *View, Reinit call*.

Under the *Info* tab machine, information about the selected virtual display. The selection of the information displayed can be adjusted (*File menu*, *Advanced Settings*, *information in the Info tab*). Practical) is the display of this configuration used QEMU-/KVM-Optionen (*show QEMU / KVM argument*).

Under the *General* tab to configure the name of the virtual machine, the emulator type (QEMU, KVM), the emulator version, the computer type (CPU architecture), the machine type, the boot sequence , the CPU type, the number of emulated CPUs and video card . Furthermore, you the keyboard layout and set the size of the memory defined. In *audio cards* to define what sound support fixed. In addition, here, the options *Full Screen Mode*, *Snapshot Mode* and *Use Local Time de-*) activated (.

The tab *HDD* images are the paths to the disk set. A maximum of four image files are included. With the green plus on the right with a new disk image *qemu-img* create. You can also overlay files based on an existing image to create. Furthermore, this is the size and image format adjustable. The button *size* can convert the image itself.

The tab *CD / DVD / Floppy* drives are the paths to the CD-/DVD-/Floppy-Images or configured.

If use the *Device Manager* enables the (*File menu*, *General settings*, *Use Device Manager by default*),

shows the *Device Manager* tab. The *riders HDD* and *CD / DVD / Floppy* are hidden. It can set parameters for the option *-drive* can be specified. The icons under *Add devices* allow the creation of new storage devices.

The tab of the *Network*, you can network settings to adjust. Under *Network cards*, one or more network cards configured. Here, the relevant type of card and the connection can be set. Desktop *Network Settings GUI* type can be considered from *Basic* to convert *native*. The Rider *Network Cards* on changes to *Network Settings*. Desktop *Connection Type* can then include the *options-net nic,-net user-net tap, net-net-socket* and *dump* choose from. Under *Additional settings* are port redirects , a TFTP prefix and a Samba share configured.

Under the *Ports* tab to debug and Expert options given. This includes the redirection of serial and parallel ports. Furthermore, Linux USB Device configured. Only on Linux, the USB devices to the guest-host systems are provided. AQEMU scans the host system to the presence of USB devices and displays them under the tab *ports, USB ports*. Requirement is the support in */ proc / bus / usb /* by the kernel.

Under the *Advanced* tab, *Additional Options* can be special options select: *RTC TD Hack , No Shutdown , No Reboot , start CPU at startup* and *Start Date* . *Additional argument* behind *QEMU* can enter options that AQEMU not supported. These options are appended in addition to those generated by AQEMU options. Will *Send Only User Arguments Without AQEMU argument* activated, only the options entered here are used. If the command line defines are completely free to *use emulator binary* is deactivated. Under the *Info* tab in the *QEMU* command line *argument* you can check. Under *QEMU / KVM Window Options*, the properties of the window affect the virtual machine. The *initial* options *Graphical Mode* is available only with PowerPC - and the Sparc -emulation.

The tab may be under *Other Hardware Virtualisation* special options for *KVM* and *KQEMU* pretend. In *VNC* with the access to the virtual machine VNC defined. Under *Optional images* are optional images , such as ether-boot image included. Under *boot Linux kernel* special Linux boot options specified.

The main window is the *File* menu, *VM*, and *Help*.

File menu

- *Create HDD Image*: Opens the *Create HDD image*.
- *Convert HDD Image*: Opens the *Convert HDD image*.
- *General Settings*: Opens the *General Settings*.
- *Advanced Settings*: Opens the *Advanced Settings*.
- *First Run Wizard Start*: Starts the Configuration Wizard AQEMU.
- *Exit*: Exits AQEMU. Still running virtual machines is a security check.

VM menu

- *New VM*
 - *Add New VM*: Creating a new virtual machine.
 - *Wizard*: Create a new virtual machine with the Wizard.
 - *VM Load From File*: load the configuration file of a virtual machine. AQEMU stores the configuration in XML files with the extension. *Aqmu* in *~ / aqemu*.
- *VM Delete*: Removes the configuration for the selected virtual machine from the list. The image files are not deleted.
- *Delete Files And VM*: Deletes the selected virtual machine with their image files.
- *Save as Template*: Opens the *Create Template*.
- *Copy*: Copies the selected virtual machine.
- *Start*: Boots the selected virtual machine.
- *Save*: Saves the state of the virtual machine as a VM snapshot with the name *aqemu_save*. This image must be present in qcow2 format and select the *snapshot* can not be activated. You can start the virtual machine in snapshot manager with the VM snapshot.
- *Pause*: pause the virtual machine fails again or continue.
- *Stop*: off the virtual machine. This can cause data loss.
- *Reset*: Performs a reboot. This can cause data loss.
- *Manage Snapshots*: Opens the *Manage VM snapshots*.
- *Show emulator Control*: Opens the *emulator control* a running virtual machine for.
- *Show QEMU / KVM Arguments*: Shows the QEMU / KVM-start options.
- *Create Shell Script*: Generates a shell script to call the selected machine.

- *Show QEMU / KVM Error Log Window*: Opens the *Error Log* to display error messages.
- *Change Icon*: Changes the icon of the virtual machine.

Help menu

- *About AQEMU*: Displays information about the program and its developers. The tab *links* interesting links listed.
- *About Qt*: Displays information about the Qt library.

General Settings window

- *AQEMU VM Folder*: It is the set list, in which the virtual machines are stored. If there is not the directory it is created.
- *Default VM Template*: This template as default template selected a.
- *AQEMU Interface Language*: This language will be adjusted.
- *GUI Icons Theme*: It is possible to select the icon themes.
- *VM Icon Size*: The sizes of the icons for the VMs are einstelbar.
- *Screenshot for use in OS icon Save Mode*: When enabled, screenshot is an icon for the virtual machine uses as.
- *Use Device Manager by Default*: The tab *Device Manager* tab replaces the *HDD* and *CD / DVD / Floppy*.
- *Emulator Control Show Mode*: These options *control* the *emulator* is configured. This allows the QEMU monitor commands are used.

Advanced Settings window

Normally in this window, no changes are necessary. Only when errors or specific requirements here should be adapted.

General tab

- *Screenshot Settings*: In the *emulator* window, you can *control* the *VM menu, Save Screenshot* Screenshot generate about. These options configure how the screenshots are saved.
- *Installed Emulators*: Here are the binaries QEMU and KVM administers the programs. It is possible to configure each multiple QEMU and KVM versions.

Advanced tab

- *AQEMU Logging*: These options determine whether and how the logging is to take place.
- *qemu-img path*: This path to *qemu-img* the specified.
- *VM Network Card MAC Address*: This specifies how MAC addresses are generated.
- *Not Use Default Audio Driver*: Here, the default audio driver or the sound system of the host will be selected between the.

Other riders

- *Execute Before / After Start / Stop QEMU*: Here are given commands that start and after you quit QEMU will run the front. In each case the full path is specified.
- *Recent CD-ROM Images Count*: This is maximum number of CD-ROM images stopped, *Connect, CD ROM, Recent Files Emulator Control* of the window showing the menu.
- *First VNC Port for Embedded Display*: Defines the VNC port for the embedded display (default = 6000).
- *Additional CD / DVD Devices*: This can be additional CD / DVD drives integrated.

Reiter information in Info Tab

The tab *information in the Info tab* is selected, the information under the *Info* tab in the main window will appear.

Window Create HDD Image

These options allow you to create virtual disks.

- *Use Base HDD Image*: Enable this option can overlay images to create.
- *New Image File Name*: This is the filename of the image defined.
- *Format*: There are those of QEMU / KVM supported image formats to choose from.
- *Size*: Except for overlay files is the size of the image file set.
- *Image Must be Encrypted*: When qcow or cqow2 format, the image is encrypted to.

Convert HDD window Image

These options allow you to convert virtual disks.

- *Input Image File Name*: This is the name of the converted image to enter.
- *Input Format*: This is the image format to convert the image to select.
- *Output Image file name*: It is the name of the destination image to enter.
- *Output format*: It is the image format of the destination image to select.
- *Must be Compressed Image*: When qcow or cqow2 format, the image is compressed to be.
- *Image Must be Encrypted*: When qcow or cqow2 format, the image is encrypted to.

Create Template window

AQEMU supports the creation of configuration templates. These templates allow complex configuration for new virtual machines are taken.

- *New VM Template Name*: Specifies the name of the template set.
- *Virtual machine file*: The configuration file of an existing virtual machine is selected as the template.
- *Use default folder for new template*: It may Directory for the template files used to be a default.
- *Creating Options*: You choose the configuration of the areas that should be taken.

VM snapshot window manager

In this window VM snapshots managed. VM snapshots are the only image format *qcow2* possible.

- *Snapshot List*: All VM snapshots are listed here.
- *Create*: Allows the creation of a VM snapshot.
- *Delete*: Allows you to delete a VM snapshot.
- *Start*: virtual machine in the state of the selected VM snapshots to start the Allows.
- *Properties*: Allows you to change the name and description.
- *Update Information*: Updates the display of the list.

menu of the VM window (emulator-Control)

These menus appear depending on the settings in the *File menu*, *General Settings*, *Control Emulator fashion show* in a separate window or embedded in AQEMU main window. These menu items, check that the instance of its term. There are commands called the QEMU monitor. This exchange is from removable media, creates screenshots and virtual machines to freeze (suspend).

VM menu

- *Save Screenshot*: Saves a screenshot in the *AQEMU Settings* defined in directory.
- *Save screenshot as*: Saves a screenshot. A file selection window.
- *Save VM*: Saves the state of the virtual machine.
- *Manage Snapshots*: Opens the *Manage VM snapshots*.
- *Commit*: The option *-snapshot* protects the storage media of change. If the virtual machine is started with this option, the command ensures *commit* the changes anyway.
- *Pause*: pause the virtual machine fails again or continue.
- *Power Off*: Turns off the virtual machine. This can cause data loss.
- *Reset VM*: Performs a reboot. This can cause data loss.
- *Qemu Monitor*: Opens a window for entering commands in the qemu monitor.
- *Quit*: Closes the *emulator control*. You can open it again from the menu *VM*, *Show Control Emulator* main window.

Connect Menu

Access to removable media and USB devices it controls these functions.

- *Floppy A*
- *Floppy B*
- *CD-ROM*
- *USB*

Send Keys menu

This menu-points can determine key combinations to the host system will be sent.

View Menu

- *Display Scaling*: Allows resizing of the window of the virtual machine.
- *Full Screen Mode*: Switch to Full Screen mode.
- *Reinit VNC*: Reinitializes the VNC access.

Help menu

- *About Emulator Control*: Displays information about the program appears.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_AQEMU "

This page has been accessed 18,114 times. This page was last updated on 10 January 2011 at 17:13 clock changed. Content is available under GNU Free Documentation License 1.2 .

libvirt, virsh, Virtual Machine Manager console virsh, virt-manager, eucalyptus

(Link to this page as [[QEMU-KVM-Buch / Management Tools / libvirt-Tools]])

<<< | # # # | >>> | English

The libvirt library

Website: <http://libvirt.org>

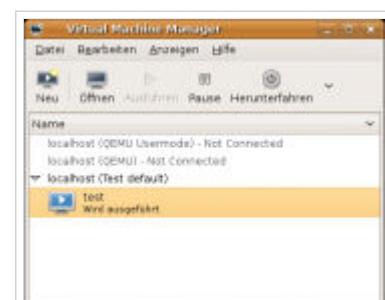
The management of virtual machines is different under the different virtualization solutions. In a heterogeneous data center, it is often impossible to control with a tool more virtualization solutions. To solve this problem, the C library developed *libvirt*. It provides standard interfaces for managing different virtualization solutions. There are drivers for QEMU, KVM, VirtualBox, VMware ESX, Xen, LXC Linux container system, OpenVZ, User Mode Linux, OpenNebula and storage systems. This is the library with their *libvirt* API provides a layer between virtualization software and management tools. Thus, the development of management tools for all virtualization solutions that support this library possible. However, not all virtualization solutions support the ability of *libvirt libvirt* or does not support all functions of any virtualization solution.

In addition to managing the virtual machines are also the management of virtual storage devices, virtual networks and devices of the host system. These configurations are stored as XML files. Among other management tools are the *virsh* (command line) and the Virtual Machine Manager (GUI). In addition to these tools for managing the virtual machines can write to their own programs. The *libvirt* library provides only the interface for the C programming language interfaces for the languages C #, Python, Perl, OCaml, Ruby and Java.

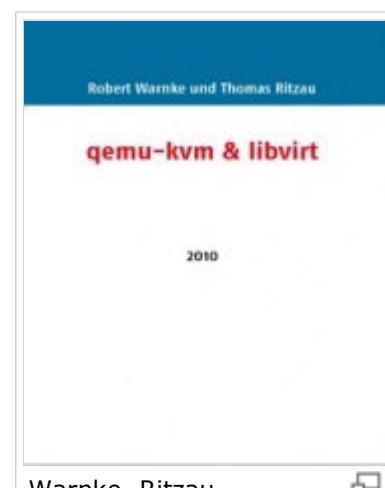
For communication between the host systems (nodes) daemon is used to *libvird*. This daemon on all nodes must be started and is higher than the API of *libvirt*. The daemon determines *libvird* local hypervisor and provides the appropriate drivers. The management tools also communicate to the daemon. These specific protocols are used. When the protocol ends with QEMU QEMU monitor. Communication can be encrypted. It supports the SASL authentication with Kerberos and SSL client certificates. The Virtual Machine Manager saves the VNC connection using SSH.

The QEMU driver supports software that uses QEMU options. In addition to QEMU and KVM includes xenner. Xenner allows Xen-guest systems running under KVM. The QEMU driver looks in the directory */usr / bin* under the QEMU binaries *qemu* and *qemu-system-**. In addition, the KVM device */dev / kvm* reviewed. For xenner is the binary */usr / bin / xenner* and the device */dev / kvm* investigated. The QEMU driver is a multi-instance driver. The connection type *system* requires privileged rights (*root*) and allows full access. The *session* connection type other hand, requires no privileged rights. The user can only manage resources for which he has access rights. That is, he can manage as virtual machines that are stored by them in their home directory. This connection type is suitable for virtualization on the desktop.

This is a management tool can connect to the driver, each represents a Uniform Resource Identifier (URI) is indicated. This indicates QEMU or a hypervisor. In addition QEMU is the connection type (*system* or *session*) input. In a remote administration in the URI to be checked and the address of the host system must be indicated. Optional is the type of connection as SSH tunnel to pretend. To connect to the hypervisor on the remote host must be running the daemon there *libvird*. Examples of URIs:



The Virtual Machine Manager is connected to the dummy hypervisor.



Warnke, Ritzau
qemu-kvm and libvirt
4. Edition 2010
ISBN: 978-3-8370-0876-0
276 pages, 27.27 EUR
Order

test: / / / default

This URI connects you with the local dummy hypervisor. This simulates a running domain. It is recommended that the first orders in this fake hypervisor try.

qemu: / / / session

This connects you to the local URI QEMU-/KVM-Hypervisor (connection type *session*) is normal as a user.

qemu + unix: / / / session

This connects you to the local URI QEMU-/KVM-Hypervisor (connection type *session*) is normal as a user.

qemu: / / / system

This URI connects you as user *root* with the local QEMU-/KVM-Hypervisor (connection type *system*).

qemu + unix: / / / system

This URI connects you as user *root* with the local QEMU-/KVM-Hypervisor (connection type *system*).

qemu + ssh: / / root@example.com / system

This URI connects you as user *root* on the node with a QEMU-/KVM-Hypervisor *example.com*. The connection is made through an SSH tunnel.

qemu + tcp: / / example.com / system

This connects you with a URI QEMU-/KVM-Hypervisor on the *example.com* node. Protection is provided with SASL / Kerberos.

xen: / / /

This URI connects you with the local Xen hypervisor. This is the default setting.

vbox: / / / session

This URI connects one as a normal user with the local VirtualBox hypervisor.

esx: / / example.com / no_verify = 1?

This URI connects you as user *root* to an ESX hypervisor on the *example.com* node. It is used for https protocol but not the certificate is verified.

gsx: / / example.com /

This URI connects you as user *root* with a VMware server.

The documentation at the URL <http://libvirt.org/uri.html> and <http://libvirt.org/drivers.html> listen to the supported values.

Installation

The Management Tools

Virtio modules

Language Bindings

oVirt

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/de/index.php/QEMU-KVM-Buch/_Managementtools/_libvirt-Tools "

This page has been accessed 23,380 times. This page was last updated on 27 September 2010 at 18:23 clock changed. Content is available under GNU Free Documentation License 1.2 .

Libvirt installation, virsh, Virtual Machine Manager console virsh virt-manager

(Link to this page as [[QEMU-KVM-Buch / Management Tools / libvirt Tools / Installation]])

<<< | # # # | >>> | English

Contents

1	installation of the library libvirt
1.1	The package
1.2	Source compile
2	Installing the Management Tools
2.1	The Software Packages
2.2	Source compile
3	Configuration
3.1	Access Rights
3.2	Network

Installing the libvirt library

As a package

From Ubuntu 8.10, there is the meta-package *ubuntu-virt-server*. It contains *kvm*, *libvirt-bin* and *openssh-server*. This meta-package is installed when you install Ubuntu the *Virtualization host* selects at (*tasksel*).

```
Host ~ $ sudo apt-get install ubuntu-virt-server ubuntu-virt-mgmt
```

Meta-standing packages none available or you need the package *libvirt-bin* is only, enter the following command line.

```
Host ~ $ sudo apt-get install libvirt-bin
```

compiling sources

Libvirt is not current package before, compile it from source. Given these packages are necessary:

```
Host ~ $ sudo apt-get install libxml2-dev libdevmapper-dev
Host ~ $ sudo apt-get install libxml + +2.6-dev libgnutls-dev
Host ~ $ sudo apt-get install libssh2-1-dev gettext autoconf
Host ~ $ sudo apt-get install libtool git-core checkinstall
Host ~ $ sudo apt-get install libnl-dev
```

For the same group is a daemon *libvirtd* necessary. For QEMU QEMU driver is the user to create. Furthermore, the current user group is incorporated into the *libvirtd*.

```
Host ~ $ sudo groupadd libvirtd
Host ~ $ sudo useradd -g libvirtd qemu
Host ~ $ sudo adduser 'id-un' libvirtd
```

It needs to get in and then log back on. If after that is still different access rights helps to reboot. Test command is one's own group membership with the *groups*.

```
Host ~ $ groups
... libvirtd
```

There are source packages for *libvirt* downloaded, unpacked and compiled. The *autogen.sh* script calls the script to *configure*. Possible options are the command *-./Configure help* appears. These options are added to the *autogen.sh* script to. More information can be found in the *INSTALL* file.

```
Host ~ $ mkdir -p ~ / source
Host ~ $ cd ~ / source
Host ~ $ wget http://libvirt.org/sources/libvirt-0.8.7.tar.gz
Host ~ $ tar xzvf libvirt-0.8.7.tar.gz
Host ~ $ cd libvirt-0.8.7
Host ~ $ ./Autogen.sh with-qemu qemu-user = - with-qemu-group = libvirtd
configure: Configuration summary configure: ===== configure: configure: Drivers configure: configure: Xen: yes configure: Proxy: yes configure: Network: yes
Host ~ $ sudo make
```

recommend software package is to generate a *check to install*. It is called instead of *make install*. A support can be obtained with *checkinstall - help*. The program provides *checkinstall* some questions that the specifications can be answered with mostly default. Clearly, the package name is to pretend as *libvirt to-self-compiled*. In the current directory the package is created and installed. On Debian / Ubuntu it lists the command *dpkg-l* and *dpkg-r* uninstall it if necessary.

```
Host ~ $ sudo checkinstall - fstrans = no - pkgname = libvirt-self-compiled
The package is built according to these specifications:
0 - Maintainer: [root @ knut]
1 - Summary: [libvirt-self-compiled]
2 - Name: [libvirt-self-compiled]
```

Alternatively, the software with *make install* to install it.

```
Host ~ $ sudo make install
```

It is the list of installed shared libraries to be updated.

```
Host ~ $ sudo ldconfig
```

The compiled library is */usr / local / lib* stored in. Tested the installation with the following command.

```
Host ~ $ virsh version
      Compiled against the library: libvir 0.8.7
      Use Library: libvir 0.8.7
      Use API: QEMU 0.8.7
      Running hypervisor: QEMU 0.13.0
```

The QEMU driver *libvirt* looks in the directory */usr / bin* under the QEMU binaries *qemu* and *qemu-system-**. The KVM driver also checks the device */dev / kvm*. The result is output with the following command.

```
Host ~ $ virsh capabilities
```

If the QEMU binaries in a different directory, an adaptation of links is possible. In this example, the binaries are in */usr / local / bin* /

```
Host ~ $ sudo ln-sf / usr / local / bin / qemu / usr / bin / qemu
```

Installing the Management Tools

The management tools need not be installed on the host running the hypervisor.

As software packages

The management tools are there for the meta package *ubuntu-virt-mgmt* with management tools, *python-vm-builder*, and *virt-manager virt-viewer*. The tools can be installed separately. The tools *virt-install* and *virt-clone* is contained in the package *python virtinst*. For the GUI *virt-manager* (<http://virt-manager.org>) packages are the *virt-manager* and *virt-viewer* to install. The installation of *virt-top* is made with the same package.

```
Host ~ $ sudo apt-get install python-virtinst virt-manager
Host ~ $ sudo apt-get install virt-viewer, virt-top
```

For the Python script *vmbuilder* need the package *python-vm-builder*.

```
Host ~ $ sudo apt-get install python-vm-builder
```

A collection of tools for *libvirt virt-goodies*, the package containing (<https://launchpad.net/virt-goodies>).

```
Host ~ $ sudo apt-get install virt-goodies
```

compiling sources

If these tools do not present as current packages, they are compiled. Management tools for compiling the following additional packages are needed.

```
Host ~ $ sudo apt-get install intltool rarian compat-
Host ~ $ sudo apt-get install python-libvirt python-gtk2
Host ~ $ sudo apt-get install python-dbus python-glade2 python-gconf
Host ~ $ sudo apt-get install python-libxml2 python-urlgrabber
Host ~ $ sudo apt-get install dbus-x11 python-gtk vnc-python-vte
Host ~ $ sudo apt-get install libgtk-vnc-1.0-dev gawk
Host ~ $ sudo apt-get install checkinstall libglade2-dev
Host ~ $ sudo apt-get install python-rsvg
```

The tools *virt-viewer*, *virt-manager virtinst* and from the site <http://virt-manager.org> downloaded, unpacked compiled and installed.

```
Host ~ $ mkdir-p ~ / source host ~ $ cd ~ / source host ~ $ wget http://virt-manager.org/download/sources/virt-viewer/virt-viewer-0.2.0.tar.gz Host ~ .
Host $ cd ~ / source
Host ~ $ wget http://virt-manager.org/download/sources/virtinst/virtinst-0.500.4.tar.gz
Host ~ $ tar xvzf *.tar.gz virtinst
Host $ cd ~ / virtinst *
Host ~ $ sudo python setup.py install
Host ~ $ sudo python setup.py sdist
Host ~ $ sudo python setup.py refresh_translations

Host $ cd ~ / source
Host ~ $ wget http://virt-manager.org/download/sources/virt-manager/virt-manager-0.8.5.tar.gz
Host ~ $ tar xvzf virt-manager-*.tar.gz
Host $ cd virt-manager *
Host ~ $ ./Configure & make
Host ~ $ sudo checkinstall -fstrans = no
The package is built according to these specifications:
0 - Maintainer: [root @ knut]
1 - Summary: [virt-manager-self-compiled]
2 - Name: [virt-manager-self-compiled]
```

Configuration

Permissions

The package *libvirt-bin* contains the *virsh* management tool. As a first test to connect to the dummy hypervisor is made. The *list* command lists running on virtual machines. The dummy Hypervisor shows by default in a fictional virtual machine.

```
Host ~ $ virsh-c test: / / / default list
Id Name Status
-----
1 test running
```

The next step is to *qemu: / session* to // test.

```
Host ~ $ virsh-c qemu: / / / session list
Id Name Status
-----
```

When using the connection type *system*, an error message when a lack of corresponding rights.

```
Host ~ $ virsh-c qemu: / / / system list
libvir: Remote error: Permission denied
Error: Connection failed to the hypervisor
```

First, check whether the daemon is running *libvirtd*.

```
Host ~ $ ps aux | grep libvirt [d]
```

If no output is to start to test the daemon manually on a different console. The *-v* option generates more verbose output.

```
Host ~ $ sudo /usr/local/sbin/libvirtd-v
```

Since the connection type requires privileged rights *system*, is used to test *sudo*.

```
Host ~ $ sudo virsh-c qemu: / / / system list
Id Name Status
-----
```

With the launch of *libvirtd* driver was even the QEMU start, access rights to the socket but do not allow use by other users.

```
Host ~ $ ls-1 /usr/local/var/run/libvirt/libvirt-sock
srwx ----- 1 root root 0 2010-01-27 17:39 /usr/local/var/run/libvirt/libvirt-sock
```

The daemon is started manually to end with [Ctrl] + [C]. The access rights by matching the files *qemu.conf* to change and *libvirtd.conf*. *Libvirt* was without the option *-prefix =* compile these files are in */usr/local/etc/libvirt/*. Before editing backups should be located.

```
Host ~ $ sudo cp /usr/local/etc/libvirt/libvirtd.conf \ /usr/local/etc/libvirt/libvirtd.conf.bak host ~ $ sudo cp /usr/local/etc/libvirt/libvirtd.conf.bak /usr/local/etc/libvirt/libvirtd.conf
```

This configuration is for the first test. A further adjustment shall be made with the well-documented configuration files. This applies particularly to the improvement of safety. The file is *libvird.conf* adjusted as follows:

```
# Libvird.conf
unix_sock_group = "libvird"
unix_sock_ro_perms = "0777"
unix_sock_rw_perms = "0770"
unix_sock_dir = "/usr/local/var/run/libvirt/"
```

The configuration file for the QEMU driver receives *qemu.conf* Content:

```
# Qemu.conf
user = "qemu"
group = "libvird"
```

The ownership of these directories should be adjusted.

```
Host ~ $ sudo chown-R qemu.libvird /usr/local/var/run/libvirt/qemu
Host ~ $ sudo chown-R qemu.libvird /usr/local/var/lib/libvirt/qemu
Host ~ $ sudo chown-R qemu.libvird /usr/local/var/cache/libvirt/qemu
```

On the second console to restart *libvird*.

```
Host ~ $ sudo /usr/local/sbin/libvird-v
```

The connection to *qemu: // system* is now successfully without *sudo*.

```
Host ~ $ virsh-c qemu: // system list
Connecting to uri: qemu: // system
Id Name Status
-----
```

To automatically start *libvird* after a reboot script is a suitable init necessary. When compiling, the init script for Red Hat distributions generated. Although one can the command to start in the file */etc/rc.local* add, a clean exit is not so reached. On Ubuntu, it is possible the init script */etc/init.d/libvirt-bin* from the package *libvirt-bin* copy and adapt. Alternatively, write your own script based on the file */etc/init.d/skeleton*.

Network

A kind of virtual machines to network is the bridging with the *bridge-utils*. The installation on Debian / Ubuntu is done with the command line.

```
Host ~ $ sudo apt-get install bridge-utils
```

To configure the network service is to stop.

```
Host ~ $ sudo invoke-rc.d networking stop
```

It is also advisable to secure the network configuration file before editing.

```
Host ~ $ sudo cp /etc/network/interfaces /etc/network/interfaces-bak
```

In this example, the interface *eth0* of the host system's IP address 192.168.1.210. The gateway has the IP address 192.168.1.1. The file */etc/network/interfaces* does the following contents:

```
# / Etc / network / interfaces
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
    address 192.168.1.210
    netmask 255.255.255.0
    gateway 192.168.1.1
```

For Briding *interfaces* is the file */etc/network/adapt*. It should be noted that the interface no longer has IP address *eth0*. The former IP address of *eth0* is assigned to the *br0* interface.

```
# / Etc / network / interfaces
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet manual
auto br0
iface br0 inet static
    address 192.168.1.210
    network 192.168.1.0
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.1
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
    bridge_maxage 12
    bridge_stp off
```

Then, the network service is restarted.

```
Host ~ $ sudo /etc/init.d/networking restart
```

If all goes well, the computer is available again under the old IP address. However, this is not the interface *eth0* addressed, but the newly created interface *br0*. To check the *ifconfig* command is used.

```
Host ~ $ ifconfig
br0 ...  inet addr: 192.168.1.210 Bcast: 192.168.1.255 Mask: 255.255.255.0 ...
eth0 ...
lo ...
```

Restarting the computer is recommended to test the correct network configuration after boot.

```
Host ~ $ sudo reboot
```

dnsmasq uses *libvirt* to host systems using DHCP to assign IP addresses. *dnsmasq* is a simple DNS and DHCP servers for small networks. The names from the local network according to the file */etc/hosts* resolved. Unknown name requests are forwarded to another DNS server and stored in the cache. In order to use *dnsmasq* to *libvirt*, is in the file */etc/resolv.conf* on the host system's IP address 192.168.122.1 as a first name server to insert. The previously existing entries for name servers are not deleted. The name resolution, it is possible to address scale virtual machines with their name on the network. In each case, a point is to be appended to the name.

```
# / Etc / resolv.conf
```

```
nameserver 192.168.122.1
NameServer ...
```

```
<<< | # # # | >>>
```

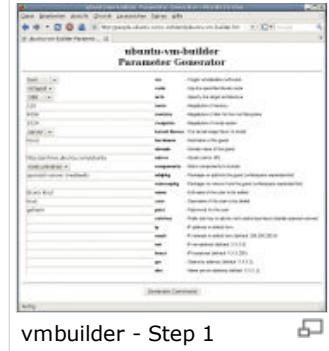
Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_libvirt-Tools/_Installation"

This page has been accessed 5599 times. This page was last updated on 27 January 2011 at 12:41 clock changed. Content is available under GNU Free Documentation License 1.2 .

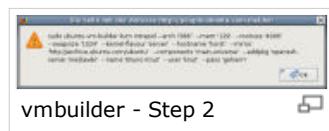
libvirt, virsh, Virtual Machine Manager console virsh, virt-manager, virt-top

(Link to this page as [[QEMU-KVM-Buch / Management Tools / libvirt Tools / The Management Tools]])

<<< | # # # | >>> | English



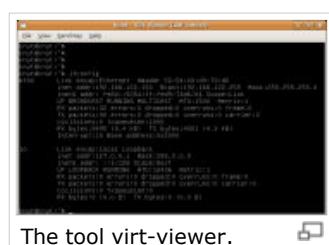
vmbuilder - Step 1



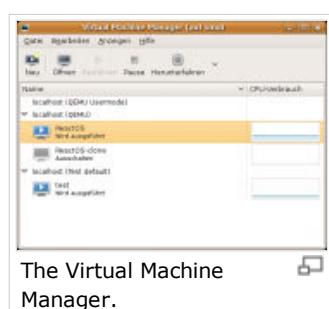
vmbuilder - Step 2



The installation of Microsoft Windows Vista on KVM with virt-install.



The tool virt-viewer.



The Virtual Machine Manager.



The Virtual Machine
Manager Wizard to create a
new virtual machine - Step 1



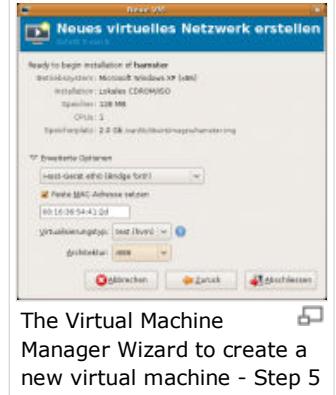
The Virtual Machine
Manager Wizard to create a
new virtual machine - Step 2



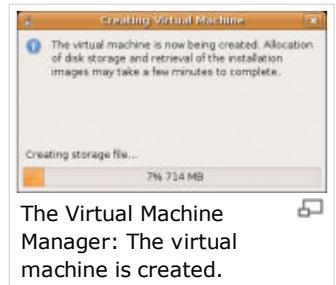
The Virtual Machine
Manager Wizard to create a
new virtual machine - Step 3



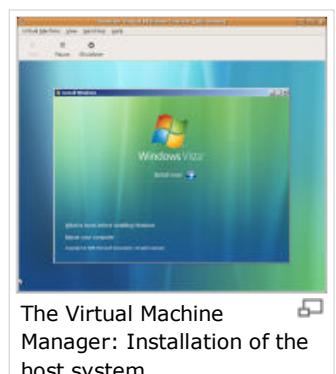
The Virtual Machine
Manager Wizard to create a
new virtual machine - Step 4



The Virtual Machine Manager Wizard to create a new virtual machine - Step 5



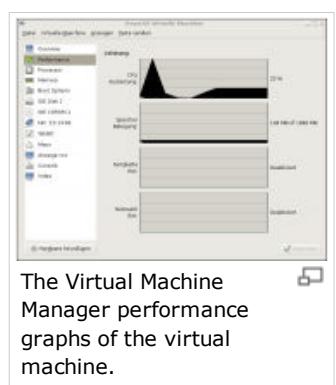
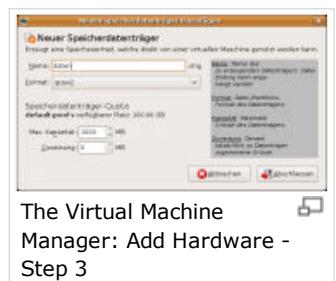
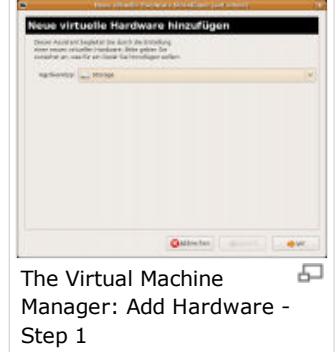
The Virtual Machine Manager: The virtual machine is created.

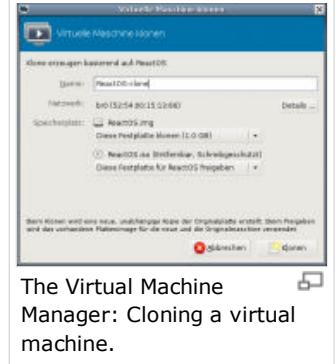


The Virtual Machine Manager: Installation of the host system.



The Virtual Machine Manager: Settings of the virtual machine.

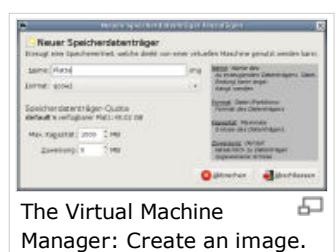




The Virtual Machine Manager: Cloning a virtual machine.



The Virtual Machine Manager: live migration.



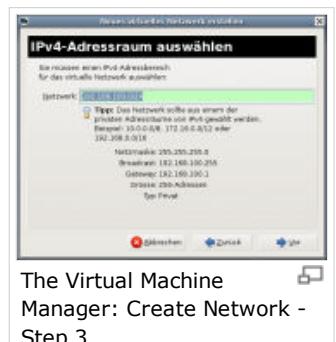
The Virtual Machine Manager: Create an image.



The Virtual Machine Manager: Create Network - Step 1



The Virtual Machine
Manager: Create Network -
Step 2



The Virtual Machine
Manager: Create Network -
Step 3



The Virtual Machine
Manager: Create Network -
Step 4



The Virtual Machine
Manager: Create Network -
Step 5

Contents

- 1 libvirt management tools
 - 1.1 virt-install
 - 1.2 virt-clone
 - 1.3 vmbuilder
 - 4.1 virsh
 - 1.5 virt-top
 - Virt-goodies 1.6
 - Virt Manager 1.7
 - 1.7.1 The main window
 - 1.7.2 Wizard "create new virtual machine"
 - 1.7.3 The Virtual Machine window
 - 1.7.4 Wizard Add new virtual hardware "
 - 1.7.5 The preferences window
 - 1.7.6 The Host Details
 - 1.7.7 Wizard "create new virtual network"
 - 1.7.8 Wizard "to add storage pool"

libvirt management tools

Command tables (man pages) to the management tools are in the Appendix (http://qemu-buch.de/d/Anhang/_libvirt).

virt-install

Virt-install the tool used for creating virtual machines, the ISO images to install the operating systems. *Virt-install* supports both text based and the graphical installer. In this case, a serial console, SDL or VNC can be used. It is also possible from installation media provided to NFS, HTTP or FTP servers to boot. The many options for this tool lists the *option-h*.

```
Host ~ # virt-install-h
```

The following example is Microsoft Windows Vista installed as a guest. For this, the installation DVD is imported. Example:

```
Host ~ $ dd if = / dev/scd0 of = vista.iso
```

With these options you start *virt-install*. It will automatically be *virt-viewer* tool (see below) is called and you install the guest system.

```
Host ~ $ sudo virt-install - connect qemu: / / / system - hvm \
  - Name fritz - ram 1024 \
  - File vista.img - file-size 10 \
  - Cdrom vista.iso \
  - Vnc \
  - Os-type windows - os-variant vista
Starting install ...
Creating storage file ... 100% |=====| 10 GB 00:00
Creating domain ... 0 B 00:00
```

virt-clone

in the package *python-virtinst* The tool included with *virt-clone* of clones enables the virtual machines. This must have driven down the virtual machine. The following example machine, the cloned *knut*. The new machine is named *Bruno* and their image is here *brunos-hd-image.qcow2*.

```
Host ~ # virt-clone - connect = qemu: / / / system-o knut \
  Bruno-n-f-hd-brunos image.qcow2
```

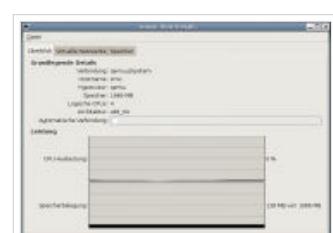
vmbuilder

The *vmbuilder* is a tool for creating virtual machines with Ubuntu JeOS as the guest system for KVM, QEMU, VMware Server, VMware Workstation, VMware Player, and Xen. Support for VirtualBox is still under development. Ubuntu JeOS (Just Enough Operating System) is a customized version of the Ubuntu Linux distribution for virtual appliances. It has to be created for each virtual machine its own directory. This directory is generated by the script a subdirectory with the necessary files. In the following example, a virtual machine with Ubuntu is created.

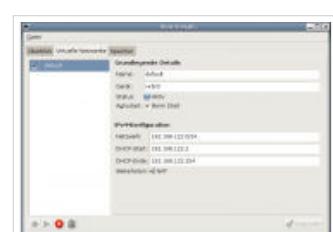
```
Host ~ $ mkdir test
```



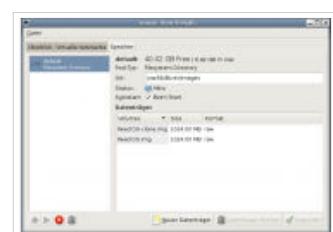
The Virtual Machine Manager: Create Network - Step 6



The Virtual Machine Manager: Information on the host system - an overview.



The Virtual Machine Manager: Information on the host system - virtual networks.



The Virtual Machine Manager: Information on the host system - memory.

```
Host ~ $ cd test
Host ~ $ sudo kvm ubuntu vmbuilder
```

To start the new virtual machine, change to the created subdirectory and calls the script *run.sh*.

```
Host ~ $ cd ubuntu-kvm
Host ~ $ ./Run.sh
```

In the following example, a virtual machine is created with Ubuntu Jaunty. The kernel should be optimized for virtual appliances for (- flavor virtual). It is bit version to install the 32 (- arch i386). The 64-bit version requires the - arch amd64 and - flavor server. With the-o Appliance is overriding the previously generated possible.

```
Host ~ $ sudo kvm ubuntu vmbuilder \
- Suite intrepid - flavor virtual - arch i386-o
```

The partitioning is with the - set *part*. As a parameter is a file named. In this example *vmbuilder.partition* is the file to create. The sizes of the partitions are specified in Mbytes. Three hyphens separate partitions on separate virtual disks. In this case, / var on a second virtual hard disk.

```
root 6000
swap 1000
---
/ Var 8000

Host ~ $ sudo vmbuilder kvm ubuntu - suite intrepid - flavor virtual \
- Arch i386-o - part vmbuilder.partition
```

The numerous options enable tools *vmbuilder* the better configuration of the virtual machine to be created. The option - help explains the options of *vmbuilder*.

```
Host ~ $ vmbuilder - help
```

More information can be obtained with the input for the hypervisor and the desired distribution.

```
Host ~ $ vmbuilder kvm ubuntu - help
```

The options of typing do not have to hand, the information contained in this web form to be entered:

<http://people.ubuntu.com/~kirkland/ubuntu-vm-builder.html>. After pressing the button *Generate Command* command line is the corresponding display. This need only copy still in the console. To be noted is that generated options for *ubuntu-vm-builder* apply. This is the predecessor of *vmbuilder*.

If you want the created virtual machine with the libvirt tools administer, so the option is - *libvirt qemu: // system* to hang. Here is an example of how to generate a virtual machine named "knut". It is thereby the configuration file / etc / libvirt / *qemu* / *create knut.xml*.

```
Host ~ $ sudo vmbuilder kvm ubuntu - suite intrepid \ - flavor virtual - arch i386-o - host name 'knut' -
```

The many options are very cumbersome, especially with repeated use. It is better to file to write the specifications and this file with the-c option must be indicated. Here is an example of a configuration file called *myappliance.cfg*.

```
[DEFAULT]
arch = i386
part = vmbuilder.partition
user = knut
name = Bruno Knut
pass = secret
[Ubuntu]
suite = hardy
flavor = virtual
addpkg = openssh-server
[Sqm]
= libvirt qemu: / / / system
```

The call *vmbuilder* reduced from themselves:

```
Host ~ $ sudo vmbuilder kvm ubuntu-c-o myappliance.cfg
```

For more information about the *vmbuilder* obtained at the URL <https://help.ubuntu.com/community/JeOSVMBuild> . a virtual machine with *libvirt* order to manage generated, it must be defined as a domain (see *virsh*).

virsh

The program serves *virsh* for managing virtual machines on the command line. It is part of the package *libvirt-bin*. With

the-c option, the specified hypervisor on which you want to connect. *Virsh* shell with a series of commands. If you want to enter only one command, it can be added immediately as an option. In this case, *list* the domains listed with.

```
Host ~ $ virsh -c qemu: / / / system list
Connecting to uri: qemu: / / / system
Id Name Status
-----
1 knut continuously
```

A remote access to another computer, in this example, *pluto*, is possible with the following options.

```
Host ~ $ virsh -c qemu + unix: / / pluto / system list
```

With the variable *VIRSH_DEFAULT_CONNECT_URI* we define the default URI. This option is the same format as used in the *connect*. If the variable as in the example set, connects to the URI *virsh test automatically: // / default* if they have no connect string is specified. To test the command is called *virsh uri*. Permanently store this variable in the file *~/.Bashrc*.

```
Host ~ $ export VIRSH_DEFAULT_CONNECT_URI = test: / / / default host ~ $ uri virsh test: / / / default
```

Virsh command is not passed, you get into the shell. The commands are listed with *help*.

```
Host ~ $ virsh -c qemu: / / / system
Connecting to uri: qemu: / / / system
virsh # help
```

To manage a virtual machine-generated, this must be defined as a domain. *Define* the command must be applied even if the XML file for the domain has changed. The configuration files of virtual machines are usually under */etc/libvirt/qemu* / or */usr/local/etc/libvirt/qemu*.

```
virsh # define / etc / libvirt / qemu / knut.xml
Knut domain from / etc / libvirt / qemu / defined knut.xml
```

To list the domains, use the command *list*. The parameter *- all* lists and inactive domains. The parameter *- inactive* lists only the inactive domains.

```
virsh # list - all
Id Name Status
-----
- Horst off
- Knut off
```

The *start* command starts the specified domain.

```
virsh # start knut
Domain knut started
```

This can be achieved with *list - test all*.

```
virsh # list - all
Id Name Status
-----
6 horst continuously
- Knut off
```

The *migrate* command moves a domain to another host. With *- live* is live migration enforced. In this example, the virtual machine to the host *pluto knut* migrated.

```
virsh # migrate - live knut qemu + tcp: / / pluto / system
```

The command can *suspend* a domain pause.

```
virsh suspend knut #
Domain knut stopped
virsh # list-all
Id Name Status
-----
6 paused knut
- Horst off
```

The command *resume* domain, the re-awakened.

```
virsh # resume knut
```

Domain knut continued

To shut down a domain, use the *shutdown* command.

```
virsh # shutdown knut
Knut domain is shut down
```

Brutal is the command to *destroy*. This moves the virtual power cord from the virtual socket.

```
virsh # destroy knut
```

With *quit*, leave this shell.

```
virsh # quit
```

The QEMU / KVM options for configuring a virtual machine with the command *domxml-from-native* to an XML configuration transformed domain. This allows you to machines that were configured with options, QEMU with the tools to manage the virtual library *libvirt*. The source format is specified with the *format*. Conversion of the QEMU *qemu* options for the *use-argv*. The configuration data is defined with *config*. Here's a simple example of *domxml-from-native*:

```
Host ~ $ echo "/ usr / bin / qemu-hda Platte.img-m 256 'q.txt'
Host ~ $ virsh domxml-from-native-argv q.txt qemu> q.xml
Host ~ $ cat q.xml
```

The command *domxml-to-native* converts an XML-domain configuration in a native host configuration. The target format is to *format* and XML file defines the *xml*. This can convert the settings from an XML configuration file into the appropriate QEMU options. Using the URL <http://wiki.libvirt.org/page/QEMUSwitchToLibvirt> find a comparison between QEMU options and libvirt configurations.

```
Host ~ $ virsh domxml-to-native-argv q.xml qemu
```

virt-top

The program *virt-top* displays statistics about the virtual machines (domains) to. The operation (options, keyboard shortcuts) similar to the *top* command. The program is called with *vir-top*. A help you get the option *virt-top - help*. A URI with *the-c* option.

```
Host ~ # virt-top-c test: / / / default
19:50:13 virt-top - i686 16/16CPU 1400MHz 3072MB 6.2% 6.2%
1 Domain, 1 active, 1 running sleeping, 0, 0 paused, 0 inactive D: 0 O: 0 X: 0
CPU: 6.2% Mem: 2048 MB (2048 MB by guests)

ID S RDRQ WRRQ RXBY TXBY% CPU% MEM TIME NAME
1 R 6.2 66.0 14641d18: 50 test
```

By default, the domains are listed. Use the following keys to change the statistics display: [1] show the CPUs. [2] show the network interfaces. [3] show the block devices. With the [q] the program is terminated. The man page gives more information about the program.

virt-goodies

Package *virt-goodies* in the program is included *vmware2libvirt*. It is used to convert VMware configuration files (. *Vmx*) the XML configuration files in *libvirt*. To convert the VMware virtual machine is shut down under and files (. *Vmx* and. *Vmdk*) to a new directory to copy the appropriate. Subsequently, these commands are used.

```
Host ~ $ vmware2libvirt-f. / File.vmx> file.xml
Host ~ $ virsh-c qemu: / / / system define file.xml
```

It converts only the configuration file. The image files remain unchanged. The conversion of images is at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware described.

Virt Manager

An example of a GUI *libvirt* on Virt is the manager. With the Virt Manager can manage virtual machines that are operated under different hypervisor types. In addition to the creation, startup, shutdown, backup and cloning it is possible to provide virtual machines (domains) with additional virtual hardware. The live migration of a virtual machine from one host to another host is possible. Furthermore, let storage pools and virtual networks set up and display performance data. The start is the Virt Manager command *virt-manager*. The options of the tool *virt-manager-h* are shown with. An overview can be found at the URL http://qemu-buch.de/d/Anhang/_libvirt.

```
Host ~ # virt-manager-h
```

With *the-c* URI is the connection to the hypervisor to specify. To test the Virtual Machine Manager with the dummy

hypervisor will be connected. Under the dummy hypervisor can be managed fictitious virtual machines.

```
Host ~ # virt-manager-c test: / / / default
```

To local QEMU-/KVM-Hypervisor (connection type *system*) to combine with the serves the following command.

```
Host ~ # virt-manager-c qemu: / / / system
```

If the *option-c* is omitted, the connection can be later in the main window *File menu, Add Connection* in the manufacture. It is the hypervisor QEMU / KVM and select the connection. If a remote connection is selected, the host name and a security protocol must be reported.

The main window

The main window shows an overview of the available virtual machines (domains). These are displayed under the hypervisor and the connection type. If you click on the small triangle next to the hypervisor, the domains are shown. Right next to the names of a domain are performance data shown as CPU consumption. In each case, under the name of the domain, you can whether it is on or off. Please click with the right mouse button on a domain name in the list opens a context menu. This menu can be commands (*Run, shutdown, pause, etc*) settle virsh. By *opening* or double-click the domain name), the *virtual machine* window (see below. The menus of the main window includes the following points:

File menu

- *Restore Saved machine ...*: Provides a previously saved virtual machine for the selected URI restore. A file selection window.
- *Add Connection ...*: Added another link to QEMU or add a hypervisor. A window appears for selection of the hypervisor and the connection. If a remote connection is selected, the host name and a security protocol must be reported.
- *Close*: Closes the main window. All other windows of the *virt-manager* to stay open.
- *Quit*: Exits the *virt-manager*.

Edit menu

- *Host Details*: Opens the *host details*.
- *Virtual Machine Details*: Opens the *virtual machine*.
- *Delete*: Deletes a selected link (URI) or a virtual machine.
- *Preferences*: Opens the *Preferences* window.

Show menu

- *Graph*: Configures the display of performance data.
 - *CPU consumption*.
 - *Hard Drives I / O*. The statistics is published on the *Edit menu, settings* are enabled in *statistics*.
 - *Network I / O*. The statistics is published on the *Edit menu, settings* are enabled in *statistics*.

Help menu

- *About*: Displays information about the program and its developers.

The icons below the menu bar have the following meanings:

- *New*: Starts the wizard for creating a new virtual machine.
- *Open*: Opens the *virtual machine*.
- *Run*: Starts the selected virtual machine.
- *Pause*: pause the virtual machine fails again or continue.
- *Shutdown*
 - *Reboot*: Reboot Runs through.
 - *Shut Down*: Shuts down the virtual machine. For paravirtualized guest systems (Xen domains) a graceful shutdown is initiated. For fully virtualized guests without a special driver made an uncontrolled exit the system.
 - *Forced off*: Pull the virtual appliance from the virtual socket.

New Virtual Machine Wizard "create"

In order to generate a new virtual machine, is chosen in the main window, the hypervisor, for example *localhost (QEMU)*. Then click on the icon *New*. It opens a wizard. This requested the following information:

1. Step:

- *Name*: The virtual machine is to be clearly identified. It may contain letters and numbers are used. At least one letter be present. Spaces are not allowed. The length must not exceed 50 characters.
- *Connection*: There are associated hypervisor available.
- *Installation Method*: select whether local CD, the network (HTTP, FTP, NFS) or PXE installed from a. For paravirtualized guest systems, the network installation (HTTP, FTP, or NFS) must be selected. It is also possible to kickstart URL. Virt Manager remembers the last five URLs for further use.

2. Step:

Depending on the chosen method of installation, the installation media or a URL to be indicated. Furthermore, the operating system type (Linux, Windows, ...) and select the appropriate version.

3. Step:

- *Memory (RAM)*: It is the amount of memory to specify.
- *CPUs*: It is the number of CPUs set.

4. Step:

In this step, you choose a place, whether a virtual disk is used. If a disk image (image file) are used, the size is set. If the entire hard drive reserved for when creating the domain, generating the virtual machine takes a little longer. The accesses to the virtual disk by the host system are then faster. There is no reservation is another disadvantage. Image only grows with the level, it comes with space problems in the host system to access unexpected errors in the host system. Alternatively, instead of an image and another to select programs. It can consider all the disks (eg / dev / sdc) mode.

5. Step:

Under *Advanced options*, which enable the network connection and the MAC address assigned in one. With *virtualization type* is only *KVM* selectable. Furthermore, the CPU architecture (*i686*, *x86_64*, *mips*, *mipsel*, *sparc* or *ppc*) to pretend.

The virtual machine window

To connect to the virtual machine can, click on their name in the main window. The window *virtual machine* with the graphics or text console. If not done, the guest system to be installed. Clicking one in this console, the mouse and keyboard in this window caught. Serves to break the key combination [Ctrl] + [Alt]. The menus support these functions:

File menu

- *Manager*: Displays the main window.
- *Close*: Closes the window. All other windows of the *virt-manager* to stay open.
- *Quit*: Exits the *virt-manager*.

Virtual Machine menu

- *Run*: Starts the selected virtual machine.
- *Pause*: pause the virtual machine fails again or continue.
- *Shutdown*
 - *Reboot*: Reboot Runs through.
 - *Shut Down*: Shuts down the virtual machine.
 - *Forced off*: Pull the virtual appliance from the virtual socket.
- *Save*: Save the memory of the virtual machine to disk and then runs them down. Restore can only state the domain with the menu item *File, Restore Saved this machine* This is not a VM snapshot. Therefore, after the virtual machine is started, the menu item *File, Restore Saved machine* ... no longer applied.
- *Cloning*: Clone the virtual machine powered down. It is the name of the new virtual machine to pretend. The network card is assigned a different MAC address. This can be adjusted in *detail*. Further cloning and sharing of storage media can be configured.
- ... *Migrate*: Migrates virtual machine to another host. This is the URI to specify the Zielsystems. If *off-line migration* disabled, migration is a live possible.
- *Screenshot record*: set a screenshot in PNG format.

Show menu

- *Console*: switching to the console view.
- *Details*: Change in the detail view for configuring the virtual hardware (see below).
- *Full screen mode*: Enables full-screen mode.
- *Resize VM*: Adjusts the size of the console window to the graphics output of the host system on.
- *Display scale*: changes the size of the display.
- *Text consoles*: Configures the text console.
- *Toolbar*: Display of icons to control the virtual machine.

Menu button to send

- It is possible to send keystrokes to the host system.

About the *View menu*, *Details*, the console version in the detail view of change. In *Overview* allows machine settings, such as *ACPI*, *APIC* and specifications for the system clock (*utc local time*,) to make. Under the heading of *performance* are performance data (CPU, memory, disk I / O, network I / A). Among the other points, the corresponding virtual hardware components are set up. Appropriate privileges are required. To virtual machine with new hardware to provide the, click on the *Add Hardware* icon. You will see the *Add New Hardware Wizard* *virtual*. For paravirtualized guest systems (Xen domains), virtual hardware components (CPUs, RAM) change on the fly.

Wizard Add new virtual hardware "

The steps of this wizard vary depending on the nature of the new hardware. The first step is prompted for the type of hardware. In this example, a virtual hard disk to be added. This type of *storage* is indicated as. The next step is to select the source. It is *block device or disk image* (Image) to choose from. For this, appropriate information shall be supplemented.

The preferences window

In this window you need to decide the behavior of the Virt Manager.

General tab

- *Enable icon in Benachrichtigungsfeld*: If this option is enabled, an icon appears on the panel of the desktop for quick access to the Virt Manager.

Reiter *Statistics*

- *Statistics Options*: Allows to configure how often the statistics display is updated and how long the history is preserved.
- *Statistics collection enable*: Enables you to statistics surveys for hard drives I / O and network I / O, they can on the *View menu, graph* are shown in the main window.

Reiter *VM Detail*

- *Consoles*: These points defined the behavior of the virtual machine console.
- *New VM*: the audio devices installed are as defined.

Reiter *Feedback*

- *Confirmations*: If this option is activated, respectively, a security query.

The Host Details

The window *details* will host the *Edit menu, host details* accessed through.

File menu

- *Manager*: Displays the main window.
- *Close*: Closes the window. All other windows of the *virt-manager* to stay open.
- *Quit*: Exits the *virt-manager*.

Reiter Overview

- *Basic details*: It is shown for the selected URI. Further information about the hypervisor, memory usage, the logical CPUs and the processor architecture will be displayed. *Automatic connection* is activated, connects the Virt Manager automatically at startup with this URI.
- *Performance*: There are the CPU utilization and memory usage graphs shown.

Reiter Virtual Networks

In this tab, virtual networks are configured. By default, a local bridge is created (*network default*). Existing networks, such *default* will be selected by clicking on the list and edited. New virtual networks are created by clicking on the icon with the plus sign. It appears Wizard to create the new virtual network (see below).

Reiter memory

This tab storage pools are configured. Existing pools, for example, *default-pool* to be selected by clicking on the list and edited. New pools are created by clicking on the icon with the plus sign. It seems the wizard to add storage pool (see below).

"New Virtual Network Wizard" to create

With this wizard, a new virtual network is created.

Example

1. *Network name*: *internal*
2. *Select IPv4 address space*: *192.168.100.0/24*
3. *DHCP scope select*: Enable DHCP, Start: *192 168 100 128*, end: *192 168 100 254*
4. *physical network with*: *isolated virtual network to physical network or pass*, target: *eth0* mode: *NAT or routed*.

Wizard Add Storage Pool "

With this wizard, a new storage pool is created. It supports the following types:

- *dir*: File System Directory
- *disk*: Physical Disk Device
- *fs*: Pre-Formatted Block Device

- iSCSI: iSCSI Target
- logical: LVM volume group
- netfs: Network Exported Directory
- scsi: SCSI host adapter

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_libvirt-Tools/_Die_Managementtools "

This page has been accessed 6199 times. This page was last updated on 27 September 2010 at 06:21 clock changed.
Content is available under GNU Free Documentation License 1.2 .

libvirt Virtio modules virtio virtio_pci virtio_ring virtio_net virtio_blk

(Link to this page as [[QEMU-KVM-Buch / Management Tools / libvirt-tools / modules Virtio]])

<<< | # # # | >>> | English

Virtio modules for guest systems

Virtio is a paravirtualization interface that only one set of drivers for guest systems to be developed (see Section *paravirtualized device drivers*). These drivers can be used by multiple virtualization solutions. This para virtual catalyzed device drivers are available for Linux guest systems with a kernel version 2.6.25. For the following examples *vmbuilder* a virtual machine created with.

```
Host ~ $ sudo kvm ubuntu vmbuilder \
- Suite intrepid - flavor virtual \
- Arch i386-o - host name 'holiday'-o \
- Mirror 'http://archive.ubuntu.com/ubuntu' \
- Components 'main, universe' - addpkg 'openssh-server' \
- Name 'I' - user 'I' - password 'secret' \
    - Libvirt qemu: // / system
```

paravirtualized network cards

In the host system to the following modules loaded: *virtio*, *virtio_pci*, *virtio_ring*, *virtio_net* and *virtio_blk*. This must KVM or QEMU with *the-net nic* and the parameter *model = virtio* start. The corresponding configuration file is to be adapted. In this example, the virtual machine *vacation* the file */etc/libvirt/qemu/urlaub.xml* <model type='virtio'>

line with the supplement, if it does not exist.

```
Host ~ $ sudo vi /etc/libvirt/qemu/urlaub.xml
<interface type='network'>
  <MAC address='52:54:00:1d:2c:86' />
  <source network='default' />
  <model type='virtio' />
</ Interface>
```

This configuration has the library *libvirt* be made known.

```
Host ~ $ sudo virsh-c qemu: // / system \
  define /etc/libvirt/qemu/urlaub.xml
Connecting to uri: qemu: // / system
Domain vacation from /etc/libvirt/qemu/defined urlaub.xml
```

Booting the guest system, for example with the *virt-manager*, and logs on. In the guest system to check if the modules were loaded.

```
Host ~ $ ssh me @ vacation.
I @ vacation.'s password secret
Guest ~ $ lsmod | grep virtio
virtio_blk 16 392 0
virtio_net 18 432 0
virtio_pci 17 668 0
virtio_ring 12 928 1 virtio_pci
virtio 14 336 3 virtio_blk, virtio_net, virtio_pci
```

paravirtualized block devices

To block devices to use a paravirtualized follows is the configuration file */etc/libvirt/qemu/urlaub.xml* how to adjust. The line is line <target dev='hda' bus='ide' /> <target dev='vda' bus='virtio' /> replace it by.

```
<disk type='file' device='disk'>
  <source file='/VMS/test/ubuntu-kvm/disk0.qcow2' />
  <target dev='vda' bus='virtio' />
</ Disk>
```

This configuration change must be made known to the *libvirt* library.

```
Host ~ $ sudo virsh-c qemu: // / system \ define /etc/libvirt/qemu/urlaub.xml
Connecting to uri: qemu: // / system
Domain vacation from /etc/libvirt/qemu/defined urlaub.xml
```

The virtual machine is started. In the host system through testing as to whether these modules were loaded.

```
Host ~ $ ssh me @ vacation.
I @ vacation.'s password secret
Guest ~ $ lsmod | grep virtio
virtio_blk 16 392 0
virtio_pci 17 668 0
virtio_ring 12 928 1 virtio_pci
virtio 14 336 3 virtio_blk, virtio_net, virtio_pci
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_libvirt-Tools/_Virtio-Module "

This page has been accessed 3317 times. This page was last updated on 7 March 2010 at 20:10 clock changed. Content is available under GNU Free Documentation License 1.2 .

Language bindings libvirt Python Perl CPAN Sys-Virt OCaml Ruby Java org.libvirt SharpLibVirt Toolbox

(Link to this page as [[QEMU-KVM-Buch / Management Tools / libvirt Tools / Language Bindings]])

<<< | # # # | >>> | English

Contents

- 1 Language Bindings
 - 1.1 C
 - 1.2 C #
 - Python 1.3
 - 4.1 Perl
 - 1.5 OCaml
 - Ruby 1.6
 - 1.7 Java



The MonoDevelop IDE.

Languages Bindings

URL: <http://libvirt.org/bindings.html>

Downloads: <http://libvirt.org/sources/>

The C library *libvirt* offers the interface for the C programming language interfaces for the languages C + +, Python, Perl, OCaml, Ruby and Java. Are required to show the corresponding binding of the programming library and A started *libvirtd*.

C

C is a procedural programming language. C is widely available for almost all computer systems. C was standardized on several occasions (C89, C95, C99, ISO-C). Most PC-/Server-Implementierungen adhere closely to the standard. A complete implementation of current standards is rare. C programs that do not include hardware-related programming can be ported to other target systems. C was developed in the early 1970s for the Unix operating system. Many other languages such as C + +, Objective-C, C #, Java, PHP and Perl are based on the syntax and other properties of C. A Wikibook on C is at the URL http://de.wikibooks.org/wiki/C_programming available. It must be installed following packages.

```
Host ~ $ sudo apt-get install gcc build-essential make
```

It tests the availability of the C compiler.

```
Host ~ $ gcc - version
gcc (Ubuntu 4.4.1-4ubuntu9) 4.4.1
```

The documentation for libvirt-Binding can be found at the URL <http://libvirt.org/html/libvirt-libvirt.html>. The corresponding package *libvirt-dev*.

```
Host ~ $ sudo apt-get install libvirt-dev
```

The following example program shows the started virtual machines. This is the file *domains.c* *show-create* with the following content.

```
# Include <stdio.h>
# Include <stdlib.h>
# Include <libvirt/libvirt.h>

static void show domain (virConnect * conn, int id)
```

```

{
    virDomain * domain;
    domain = (conn, id) virDomainLookupByID;
    printf ("% s:% d \ n", domain virDomainGetName (domain), virDomainGetID ());
}

static void show domains (virConnect * conn)
{
    int number_of_domains, id, * ids;
    number_of_domains virConnectNumOfDomains = (conn);
    ids = malloc (number_of_domains * sizeof (int));
    if (ids)
        Return;
    virConnectListDomains (conn, ids, number_of_domains);
    for (id = 0; id < number_of_domains; id + +)
        show domain (conn, ids [id]);
    free (ids);
}

int main ()
{
    virConnect * conn;
    virConnectOpen conn = ("test: / / / default");
    if (conn)
        return -1;
    show domains (conn);
    virConnectClose (conn);
    return 0;
}

```

The source code is to compile and run the program.

```

Host ~ $ gcc-o lvirt-show-show-domains domains.c
Host ~ $ . / Show-domains
test: 1

```

C

The object-oriented language C # (C Sharp) was developed by Microsoft for the. NET strategy. C # support is based on concepts of Java, C ++, SQL, Delphi and C and both the development of language-independent. NET components and COM components for use with Win32 applications. A Wikibook about C # is at the URL http://de.wikibooks.org/wiki/Programmierkurs_C-Sharp available. It will need the following packages.

```

Host ~ $ sudo apt-get install mono-complete mono-gmcs
Host ~ $ sudo apt-get install mono-devel monodevelop

```

It tests the availability of the compiler.

```

Host ~ $ gmcs - version
Mono C # compiler version 2.4.2.3

```

The *MonoDevelop* IDE is invoked as follows.

```

Host ~ $ monodevelop &

```

Using the URL <http://svn.i-tux.cz/listing.php?repname=SharpLibVirt> find the libvirt-binding.

Python

Website: <http://www.python.org>

Python is a high-level programming. Features include high performance and clear syntax. The standard library included is extensive. Unusually, the use of indentation on the limitation of blocks. A Wikibook is under the URL <http://de.wikibooks.org/wiki/Python-Programmierung> available. Python is available on most Linux distributions have a package installed *python*. With the following command tests to availability.

```
Host ~ $ python - version
Python 2.6.4
```

The documentation for the libvirt-binding is under the URL <http://libvirt.org/python.html> . You need the package *python-libvirt*.

```
Host ~ $ sudo apt-get install python-libvirt
```

The example *show-domains.py* shows running domains.

```
import libvirt
conn = libvirt.open ('test: / / / default')
for id in conn.listDomainsID () :
    domain = conn.lookupByID (id)
    print "% s:% s%" (domain.name () domain.info () [0])
```

The script is called as follows.

```
~ $ Python-show host domains.py
test: 1
```

Perl

Website: <http://www.perl.org>

Perl is a free, platform-independent, interpreted scripting language. Perl was originally developed for the manipulation of text files. The strengths of Perl, the processing of data streams of various news sources, and to prepare documents using regular expressions. There are many freely available modules that are placed in a central location (CPAN) is available. CPAN (Comprehensive Perl Archive Network) is an online repository (<http://www.cpan.org>) for Perl modules. A Wikibook about Perl is available at URL <http://de.wikibooks.org/wiki/Perl-Programmierung> available. Perl is available on most Unix-like systems (* BSD, Solaris, Mac OS X, Linux) pre-installed. With the following command tests to availability.

```
Host ~ $ perl - version
This is perl, v5.10.0 built for x86_64-linux-gnu-thread-multi
```

Many Perl modules are available as software packages (*lib *- perl*). If there is not a Perl module as a package that you can install it with CPAN. This package is the *perl modules* needed.

```
Host ~ $ sudo apt-get install perl-modules
```

As *root* you call *cpan*.

```
Host ~ $ sudo-i
Host ~ # cpan
```

The first call to configure *cpan* are some questions to put. The default answers are acceptable. When asked about the mirror server is a server near you is to be selected. The configuration is Debian / Ubuntu in / *etc* / *perl* / *CPAN* / *Config.pm* saved. After successful configuration of CPAN prompt waits for commands.

```
cpan> quit
```

The Perl module *Sys::Virt* provides the Perl bindings for *libvirt*. Documentation is available at URL <http://search.cpan.org/dist/Sys-Virt/lib/Sys/Virt.pm> available. Is there no such package to install this module via CPAN. With the following command searches to the Perl module *Sys::Virt*.

```
Host ~ # cpan
cpan> i / Sys:: Virt /
```

Installing a Perl module is to *install*.

```
cpan> install Sys:: Virt
cpan> quit
Host ~ # exit
```

With the following command tests to the successful installation of the Perl module. If no output appears, everything is correct.

```
Host ~ $ perl-MSys:: Virt-e exit
```

For example, create the file *show-domains.perl* with the following content.

```
# / Usr / bin / perl-w
use Sys:: Virt;
my $ con = Sys:: Virt-> new (uri = "test: / / / default");
my @ domains = $ con-> list_domains;
for my $ domain (@ domains) {
    printf "% s:% s \ n",
        $ Domain-> get_name, $ domain-> get_info-> {state};
}
```

The file is made executable and called.

```
Host ~ $ chmod + x domains.perl show host ~ $. / Show-domains.perl test 1
```

OCaml

Website: <http://caml.inria.fr>

OCaml (Objective CAML) is based on the ML family of languages (Meta Language) programming language. In addition to the functional and imperative features of ML OCaml supports object-oriented concepts. OCaml has a compiler for the generation of byte code and machine code. OCaml is available for many platforms, including Unix and Windows. How you can find the URL for OCaml <http://www.ocaml-tutorial.org/de>. It will need the following packages:

```
Host ~ $ sudo apt-get install ocaml ocaml-interp
```

It tests the availability.

```
Host ~ $ ocaml-version
The Objective Caml toplevel, version 3.11.1
Host ~ $ ocamlopt-config
version: 3.11.1
```

The libvirt-Binding can be found at the URL <http://libvirt.org/ocaml/>. It will need the following packages:

```
Host ~ $ sudo apt-get install libvirt-ocaml-libvirt-ocaml-dev
```

For example, create the file with the following content *list_domains.ml*.

```
module C = Libvirt.Connect
module D = Libvirt.Domain
let name = "test: / / / default"
let conn = C.connect_READONLY ~ name ()
open Printf
let doms =
let n = C.num_of_domains in conn
let ids = n C.list_domains in conn
let domains = Array.map (D.lookup_by_id conn) ids in
```

```
Array.iter (
  fun dom -> printf "%s %d \n%"! (D.get_name dom) (D.get_id dom)
) Domains;
```

The source code is to compile and run the program.

```
Host ~ $ ocamlopt-I + libvirt mllibvirt.cmxa list_domains.ml \
-O list_domains
Host ~ $. / List_domains
test 1
```

Ruby

Website: <http://ruby-lang.org>

Ruby is a powerful, object-oriented language. All data types, including numbers and strings are objects. Ruby was designed as a *multi-paradigm language*. That is, it is open to other programming paradigms to the developer to create its programs to use. A quick start is at the URL <http://www.ruby-lang.org/de/documentation/quickstart/>. A Wikibook about Ruby is at the URL <http://de.wikibooks.org/wiki/Ruby-Programmierung> available. It must be installed following packages.

```
Host ~ $ sudo apt-get install ruby-full rubygems rake irb
```

It tests the availability.

```
Host ~ $ ruby -v
ruby 1.8.7 (2008-08-11 patchlevel 72) [x86_64-linux]
Host ~ $ irb
irb (main):> quit 001:0
```

The libvirt-Binding can be found at the URL <http://libvirt.org/ruby/>. The accompanying documentation is licensed under <http://www.libvirt.org/ruby/api/> published. It is the package *libvirt-ruby* install.

```
Host ~ $ sudo apt-get install libvirt-ruby
```

For the first steps to provide *interactive ruby* shell to the. With *require* you have to use the library. Then the object is defined *connect*. *Puts* serves to output information to the screen. In this case, the document is an XML output that properties of the currently associated describes the hypervisor (see *virsh capabilities*).

```
Host ~ $ irb
irb (main): 001:0> require 'libvirt'
=> True
irb (main):> connect = Libvirt::open("test: / / / default")
=> # <Libvirt::Connect:0xb77c4554>
irb (main): 003:0> puts connect.capabilities
...
irb (main):> quit 004:0
```

For a sample script create the file *show-domains.rb* this content with.

```
# /usr/bin/env ruby
require 'libvirt'
connect = Libvirt::open("test: / / / default")
domains = []
domains.each do | id |
  domain = connect.lookup_domain_by_id(id)
  puts "# {domain.name}: # {} domain.id"
end
```

The file is made executable and called.

```
Host ~ $ chmod +x show-domains.rb
Host ~ $ ./Show-domains.rb
test: 1
```

Java

Web site: <http://java.sun.com>

Java is an object-oriented, platform-independent programming language. Java programs are translated into bytecode and then executed in the Java runtime environment. Whose main component is the Java Virtual Machine. Java is a registered trademark of Sun Microsystems (now Oracle Corporation). Sun offers the Java VM for Linux, Solaris and Windows. Other manufacturers have their Java VM for your platform certification, such as the Apple for Mac OS X. A Wikibook on Java under the URL <http://de.wikibooks.org/w/index.php?title=Java&oldid=1000000> available. In addition to the package *sun-java6-jdk* packages are the *ant*, *libjna-java* and *junit* necessary.

```
Host ~ $ sudo apt-get install sun-java6-jdk java-junit ant libjna
Host ~ $ sudo update-alternatives - config java
```

It tests the availability.

```
Host ~ $ javac -version
javac 1.6.0_16
Host ~ $ ant -version
Apache Ant version 1.7.1 compiled on October 19 2009
```

The documentation for the libvirt-Binding can be found at the URL <http://libvirt.org/sources/java/javadoc/>. The files of the libvirt-binding, you charge by the URL <http://libvirt.org/sources/java/> down and starts the build process with *ant*. Then the built jar archive should be copied into the directory containing the Java libraries. The CLASSPATH of the current directory and the directory containing the files and *libvirt.jar* must face. The variable LD_LIBRARY_PATH points to the path of the file *libjnijnidispatch.so*.

```
Host ~ $ mkdir -p ~ / source
Host ~ $ cd ~ / source
Host ~ $ wget \
http://libvirt.org/sources/java/libvirt-java-0.4.2.tar.gz
Host ~ $ tar xzvf libvirt-java-0.4.2.tar.gz
Host ~ $ cd libvirt-java-0.4.2
Host ~ $ ant build
Host ~ $ sudo cp target/libvirt-0.4.2.jar /usr/share/java/
Host ~ $ sudo ln -s /usr/share/java/libvirt-0.4.2.jar \
    /usr/share/java/libvirt.jar
Host ~ $ export CLASSPATH =.: /usr/share/java/*
Host ~ $ export LD_LIBRARY_PATH = /usr/lib/jni/
```

For example *ShowDomains.java* is the file the following content to mess with.

```
import org.libvirt.*;
public class show {domains
    public static void main (String [] args) {
        Connect conn = null;
        try {
            conn = new Connect ("test: / / / default", true);
        } Catch (LibvirtException e) {
            System.out.println ("Exception caught:" + e);
            System.out.println (e.getError ());
        }
        try {
            int [] domains = conn.listDomains ();
            for (int i = 0; i < Domains.length; i + +)
            {
```

```
        Domain = dom conn.domainLookupByID (domains [i]);
        System.out.println (dom.getName () + ":" + dom.getID ());
    }
} Catch (LibvirtException e) {
    System.out.println ("Exception caught:" + e);
    System.out.println (e.getError ());
}
}
```

This file is compiled and called.

```
Host ~ $ javac ShowDomains.java & & java show domains
test: 1
```

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_libvirt-Tools/_Languages_Bindings "

This page has been accessed 3732 times. This page was last updated on 27 September 2010 at 06:21
clock changed. Content is available under GNU Free Documentation License 1.2 .

oVirt, libvirt

(Link to this page as [[QEMU-KVM-Buch / Management Tools / libvirt tools / oVirt]])

<<< | # # # | >>> | English

oVirt

Website: <http://ovirt.org>

oVirt allows the management of virtual machines through a web interface. This company Red Hat open source project sponsored to use the library *libvirt*. oVirt provides the hardware resources (CPUs, RAM and disk storage) of several hosts of the pool available to virtual machines. The assignment of resources to the pools is flexible. Booting the hosts (oVirt nodes) are each from a CD, a USB stick or PXE. Local hard disks are not necessary. The oVirt-node image contains a minimal Fedora with kernel-based Virtual Machine and the *libvirt* library. The processors that hosts must support hardware virtualization.

The current version of the management system is available as a special KVM appliance available. Therefore, for this virtual machine, a host system with hardware virtualization is necessary. Continue to be required for the productive use of at least two network cards. The storage is provided via iSCSI or NFS. For a separate iSCSI network is set up.

Installing the Management Appliance

It is necessary to ensure that support for hardware virtualization is enabled in the BIOS of the computer (example):

```
Advanced
CPU Configuration
Virtualization Technology [Enabled]
```

On Linux, it is determined whether the processors used the hardware virtualization (hardware virtual machine - HVM) support.

Host ~ \$ **grep "vmx" / proc / cpufreq**

If lines *vmx* output is Processor type Intel VT.

Host ~ \$ **grep "svm" / proc / cpufreq**

When rows *svm* plotted, it is of type processors AMD-V. No output will not support hardware virtualization. oVirt can not be used.

oVirt For at least two network cards are required. The operating system is Fedora to install version 11 (64-bit) on that host. It is set to U.S. English as the language system. is given in the file */etc/environment* *LANG = C* write the row. It is to update the system.

```
Host ~ # yum check-update
Host ~ # yum update
```

A reboot is recommended.

```
Host ~ # reboot
```

After updating the system repository, the following RPM packages from *rawhide* to install.

```
Host ~ # yum install - enablerepo rawhide = \
          ovirt ovirt-server-installer-image ovirt ovirt-node-node-image-pxe
```

oVirt the installer to call.

```
Host ~ # ovirt-installer
This installer will configure the installation based on a series ovirt
of questions. When complete, you will be asked to install or oVirt
do the installation manually. Would you like to continue? | Y | Return
```

```
SELinux enforcing, would you like to set it permissive? | Y | Return
Setting SELinux permissive
```

Below are the active networking devices, if there are any missing devices

```
Ensure they are active and have an ip address before running the installer
mac address interface ip address
00:24: e8: 68:02:74: eth0: 172.16.9.202
```

```
Enter the interface for the Guest network: | eth0 | Return
Enter the admin interface for the network (this may be the same as the Guest network interface): | eth0 | Return
```

```
Enter the hostname of the oVirt management server (example: management.example.com): management.example.com
```

```
The following DNS servers were found:
nameserver XXX.XXX.XXX.XXX
nameserver XXX.XXX.XXX.XXX
```

If your dns server container above the A records for your management server select "y" otherwise select "n" and a dns server

Use this system's dns servers? **Y**

an in with a DHCP server should be tested if oVirt network, is the following question must be answered with **y**. A wrongly configured DHCP server provides network problems.

Does your network admin already have dhcp? **Y**

Do you have a cobbler server already that you wish to use? **No**

We will setup a cobbler instance, please provide the following information

Enter your username cobbler: **cobbleruser**

Enter your cobbler user password: *********

Confirm your cobbler user password: *********

Enter a password for the postgres account ovirt: *********

Confirm your password postgres ovirt *********

Enter your realm name (example: example.com): **ovirt.example.com**

NOTE: The following password will then be your ovirtadmin password for the web management login

Enter an administrator password for FreeIPA: *********

Confirm your FreeIPA admin password:

To start the installation run: ace install ovirt

oVirt by a bug in the package dependencies of the following steps are necessary.

```
Host ~ # rpm-qi 389-ds-base host ~ # rpm-e 389-ds-base - nodeps host ~ # yum install perl-Mozilla-LDAP nss-tools-tools mozld
```

Then installed oVirt.

```
Host ~ # ace-d-1 / tmp / install ovirt ace.log  
Command Failed with return code 1
```

Ignore error?

- https://bugzilla.redhat.com/show_bug.cgi?id=518544

Post Install

```
Host ~ # cobbler distro add - name F11-i386 - kernel = / tmp / isolinux / vmlinuz - initrd = / tmp / isolinux / initrd.img
```

URL: <https://yourserver/cobbler/>

Im_Aufbau

<<< | # # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_libvirt-Tools/_oVirt"

This page has been accessed 8570 times. This page was last updated on 27 September 2010 at 06:22 clock changed. Content is available under GNU Free Documentation License 1.2 .

libguestfs guest virt-cat fish virt-edit virt-df, virt-p2v

(Link to this page as [[QEMU-KVM-Buch / Management Tools / libguestfs]])

<<< | # # # | >>> | English

Under construction

Contents

- 1 The library libguestfs
 - 1.1 Installation
 - 1.1.1 Fedora, RHEL with EPEL
 - 1.1.2 Compile
 - 1.2 Linking

The Library libguestfs

Website: <http://libguestfs.org>

The C library *libguestfs* (Open Source) is an API for working with virtual media (images), the virtual machines are used to. With the tools in this library (virt-cat, virt-edit, virt-df, virt-p2v) can be in host systems edit files, compile statistics and the file systems and change registry entries (Microsoft Windows). Other features are: backup, cloning, and migration. Support will include the file systems ext2/3/4, btrfs, FAT and NTFS, LVM, and different image formats, such as *qcow2* and *vmdk*. It allows virtual machines, edit the *libvirt* library to be managed (see Section *Management tools*). The library uses *libguestfs* features Kernel-based Virtual Machine. The C library provides *libguestfs* APIs for C, C++, OCaml, Perl, Python, Ruby, Java, Haskell, C# and shell scripts.

Installation

Fedora, RHEL with EPEL

Under Fedora since version 11 and RHEL with REPEL (Extra Packages for Enterprise Linux) to install the necessary packages with these commands.

```
Host ~ $ su -
Host ~ # yum install '*' * guestfs'
```

The installation will be tested with an actual image.

```
Guest host ~ # fish - ro-a disk.img
```

Compiling

To compile the following software is required:

- QEMU / KVM from version 0.10 and there must *vmchannel* support.
- The package *febootstrap* version 2.7.
- The package *fakeroot*.
- The package *fakechroot* version 2.9.
- ? XDR, rpcgen (on Linux these are provided by glibc)
- The program *mksquashfs* of package *squashfs-tools*.
- Either the package or the package *genisoimage* *mkisofs*.
- ? (Optional) *hivex>* = 1.2.1 to build support Windows Registry
- Optional fuse is needed for the build of fuse modules.
- Optionally Augeas (<http://augeas.net>) are required.
- To generate the documentation, programs, and the *pod2man* *pod2text* of the Covenant *perldoc* necessary.
- For a nicer command line in *guest fish* can be installed *readline* optional.
- Optional is to validate the RELAX NG schema from the *virt-inspector* requires *xmllint*.
- Optionally, the packets *libocamlnet-ocaml-bin* and *library xml-light* for the Ocaml-Binding installed (<http://tech.motion-twin.com/xmllight.html>).
- Optionally, a local Fedora mirror is configured.
- Optional Perl for Perl bindings to install.
- Optional Python bindings for the Python is installed.
- Optional Ruby for Ruby bindings to install.
- Optional Java for the Java bindings to install.
- GHC is an option to install the Haskell bindings.
- Optional Perl modules *XML::XPath* and *Sys::Virt-Support* for the libvirt *virt-inspector* in Ireland to install. .
- Optional but recommended the installation of *perl-libintl*.

Installing QEMU / KVM is at the URL <http://qemu-buch.de/d/Installation> described.

```
Host ~ $ qemu - version
QEMU PC emulator version 0.12.4, Copyright (c) 2003-2008 Fabrice Bellard
```

Packages *debootstrap* and *debirf*.

```
Host ~ $ sudo apt-get install debootstrap debirf
Host ~ # debootstrap - help
Host ~ $ debirf help
```

For the packages *fakeroot* *fakeroot* and install *fakeroot-ng*.

```
Host ~ $ sudo apt-get install fakeroot fakeroot-ng
Host ~ $ fakeroot - version
fakeroot version 1.12.1
```

The package must *fakechroot* in the present at least Version 2.9.

```
Host ~ $ sudo apt-get install fakechroot
Host ~ $ fakechroot - version
fakechroot Version 2.9
```

Fakechroot is not available in version 2.9, it is compiled.

```
Host ~ $ mkdir-p ~ / source host ~ $ cd ~ / source host ~ $ wget \ http://ftp.debian.org/debian/pool/main/f/fakechroot/fakechroot_2.9.orig.tar.gz host
```

? XDR, rpcgen (on Linux these are provided by glibc)

```
Host ~ $ xdrmem_create
?
```

```
Host ~ $ sudo apt-get install libc6 libc6-dev
```

Host ~ \$ **rpctgen**

Package squashfs-tools

```
Host ~ $ sudo apt-get install squashfs-tools
Host ~ $ mksquashfs version
mksquashfs version 3.3 (10/31/2007)
```

Package genisoimage

```
Host ~ $ sudo apt-get install genisoimage
Host ~ $ genisoimage version
genisoimage 1.1.9 (Linux)
```

Optional: hivex - Windows registry hive "extraction library

- <http://libguestfs.org/hivex.3.html>
- <http://packages.debian.org/de/source/sid/hivex>

FUSE

```
Host ~ $ sudo apt-get install-dev libfuse libfuse2
```

Augeas

```
Host ~ $ sudo apt-get install libaugeas augeas-dev-tools
```

Readline

```
Host ~ $ sudo apt-get install libreadline5-dev readline-common
```

xmllint

```
Host ~ $ sudo apt-get install libxml2-utils
Host ~ $ xmllint-version
xmllint: using libxml version 20 632
```

OCaml bindings

```
Host ~ $ sudo apt-get install ocaml ocaml-interp ocaml-libocamlnet-bin
Host ~ $ sudo apt-get install libvirt-ocaml-libvirt-ocaml-dev libxml-light-ocaml-dev
```

It tests the availability.

```
Host ~ $ ocamlopt-version
The Objective Caml toplevel, version 3.11.1
Host ~ $ ocamlopt-config
version: 3.11.1
```

Optional: local Fedora mirror

Python Bindings

```
Host ~ $ python - version
Python 2.6.4
```

```
Host ~ $ sudo apt-get install python-libvirt
```

Ruby bindings

```
Host ~ $ sudo apt-get install ruby-full rubygems rake irb
```

It tests the availability.

```
Host ~ $ ruby - version
ruby 1.8.7 (2008-08-11 patchlevel 72) [x86_64-linux]
Host ~ $ irb
irb (main):> quit 001:0
```

Java Bindings

```
Host ~ $ sudo apt-get install sun-java6-bin sun-java6-jre sun-java6-jdk
Host ~ $ sudo update-java-alternatives - jre - set java-6-sun
Host ~ $ export JAVA_HOME = "/usr/lib/jvm/java-6-sun"
```

It tests the availability.

```
Host ~ $ javac-version
javac 1.6.0_20
```

Optionally you need for the GHC Haskell bindings.

```
Host ~ $ sudo apt-get install haskell-utils ghc6
```

Perldoc package

```
Host ~ $ sudo apt-get install perl-doc
Host ~ $ pod2man - help
Host ~ $ pod2text - help
```

Optional XML, the Perl Module:: XPath, Test:: More, ExtUtils:: MakeMaker, Pod:: Usage, Getopt:: Long Sys:: Virt, Data:: Dumper, Locale:: text domain, XML:: Writer , Win:: Hivex and Win:: Hivex:: Regedit needed. Many Perl modules are available as software packages (*lib *- perl*).

```
Host ~ $ sudo apt-get install libextutils-autoinstall-perl
Host ~ $ sudo apt-get install build-perl-libmodule libtest-pod-perl
Host ~ $ sudo apt-get install libxml-writer-perl libxml-xpath-perl
```

?

```
Host ~ $ sudo apt-get install perl-libwine-hivex
```

If there is not a Perl module as a package that you can install it with CPAN. This package is the *perl modules* needed.

Perl bindings

```
Host ~ $ sudo apt-get install perl-modules
```

As root you call `cpan`.

```
Host ~ $ sudo -i
Host ~ # cpan
```

The first call to configure `cpan` are some questions to put. The default answers are acceptable. When asked about the mirror server is a server near you is to be selected. The configuration is Debian / Ubuntu in `/etc/perl/CPAN/Config.pm` saved. After successful configuration of CPAN prompt waits for commands.

```
cpan> quit
```

The Perl module `Sys::Virt` provides the Perl bindings for `libvirt`. Documentation is available at URL <http://search.cpan.org/dist/Sys-Virt/lib/Sys/Virt.pm> available. Is there no such package to install this module via CPAN. With the following command searches to the Perl module `Sys::Virt`.

```
Host ~ # cpan
cpan> i / Sys::Virt /
```

Installing a Perl module is to *install*.

```
cpan> install Sys::Virt
cpan> install Win::Hivex
cpan> install Win::Regedit
cpan> quit
Host ~ # exit
```

With the following command tests to the successful installation of Perl modules. If each issue: no, everything is correct.

```
Host ~ $ perl-MXML::XPath-e exit
Host ~ $ perl-MXML::Writer-e exit
Host ~ $ perl-MTest::More-e exit
Host ~ $ perl-MExtUtils::MakeMaker-e exit
Host ~ $ perl-MLocal::Text-Domain-e exit
Host ~ $ perl-MLocal::Dumper-e exit
Host ~ $ perl-MData::Long-e exit
Host ~ $ perl-MGetopt::Usage-e exit
Host ~ $ perl-MPod::Virt-e exit
Host ~ $ perl-MSys::Hivex-e exit
Host ~ $ perl-MWin::Hivex-e exit
Host ~ $ perl-MWin::Regedit-e exit
```

(Optional, but highly recommended) perl-libintl perl code for translating.

```
Host ~ $ sudo apt-get install perl-libintl
```

Collect and compile from source:

```
Host ~ $ mkdir -p ~ / source
Host ~ $ cd ~ / source
Host ~ $ wget http://libguestfs.org/download/libguestfs-1.5.0.tar.gz
Host ~ $ tar xvzf libguestfs-1.5.0.tar.gz
Host ~ $ cd ~ libguestfs-1.5.0
Host ~ $ less README
Host ~ $ ./Configure help
Host ~ $ ./Configure with-repo = sarge
...
Daemon ..... yes
Appliance ..... yes
QEMU ..... / Usr / bin / qemu
OCaml bindings ..... yes
Perl bindings ..... yes
Python bindings ..... yes
Ruby bindings ..... yes
Java bindings ..... yes
Haskell bindings ..... yes
virt-inspector ..... yes
virt-* tools ..... yes
supermin appliance ..... yes
FUSE filesystem ..... yes
Host ~ $ make
Host ~ $ make check
Host ~ $ sudo make install
```

Problems

```
make
...
+ Debif make-debian n
Loading profile 'debian' ...
debif: clearing old debif root ...
debif: creating debif root ...
I: Retrieving Release
E: unknown location / dists / sarge / Release
make [2]: *** [.. / initramfs / fakeroot.log] Error 1
make [2]: Leaving directory `/home/ich/source/libguestfs-1.5.0/appliance'
make [1]: *** [all-recursive] Error 1
make [1]: Leaving directory `/home/ich/source/libguestfs-1.5.0'
make: *** [all] Error 2
```

Links

- <http://libguestfs.org/FAQ.html>
- <http://rwmj.wordpress.com/2010/02/15/ubuntu-9-10-packages-for-libguestfs/>
- <http://www.annexia.org/tmp/debian/>

<<< | # # | >>> <http://qemu-buch.de>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_libguestfs "

This page has been accessed 2560 times. This page was last updated on 23 July 2010 at 10:51 clock changed. Content is available under GNU Free Documentation License 1.2 .

openQRM - Cloud computing, systems management for heterogeneous data center, Xen KVM VMware ESXi, howto

(Link to this page as [[QEMU-KVM-Buch / Management Tools / OpenQRM]])

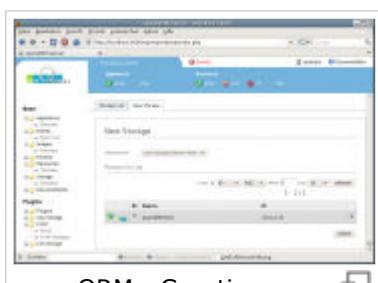
<<< | # # # | >>> | English



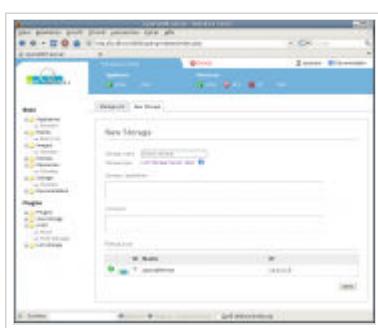
openQRM under a virtual machine KVM.



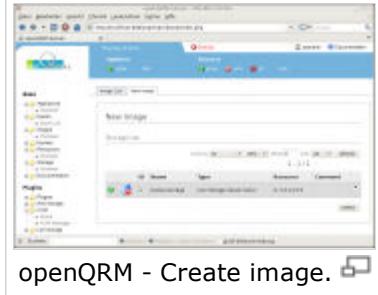
Activate the plugin in openQRM installed web interface.



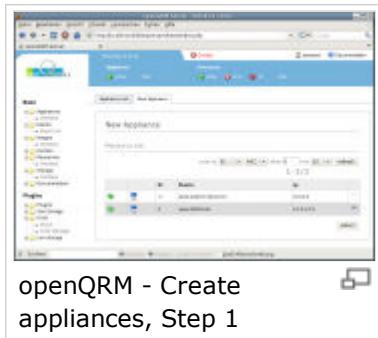
openQRM - Creating Storage, Step 1



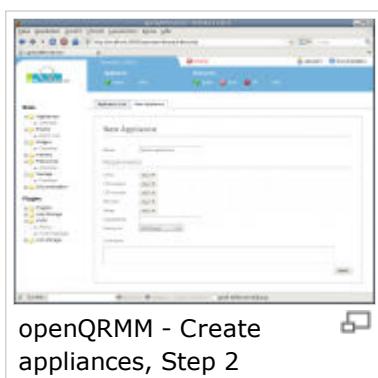
openQRM - Creating storage, step 2



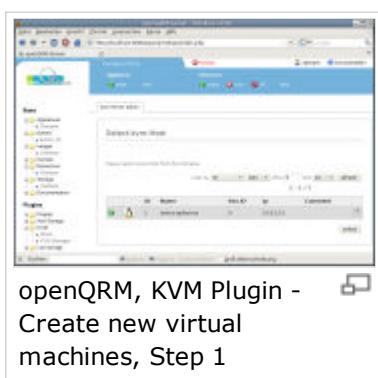
openQRM - Create image.



openQRM - Create appliances, Step 1



openQRMM - Create appliances, Step 2



openQRM, KVM Plugin - Create new virtual machines, Step 1

openQRM

Website: <http://www.openqrm.com>

openQRM a developed since 2006 as a versatile open source systems management suite. With a central management console (web interface), the management of a heterogeneous data center is possible. The standard features include software distribution, maintenance of system images, and monitoring with Nagios. Further the Admin Tool Set and manage high-availability systems. OpenQRM can be both physical and virtual machines (VMware Server, VMware ESX, Xen, Linux-VServer, Qemu and Kernel-based Virtual Machine) manage. OpenQRM runs on Linux and can be extended by plugins.

test in a virtual machine

Download: http://sourceforge.net/project/showfiles.php?group_id=153504 (32-bit)

For a first test openQRM can be installed in a virtual machine. In this example, the *vmbuilder* a virtual machine with Ubuntu 8.04 JeOS created with and under the Kernel-based Virtual Machine operated (see http://qemu-buch.de/d/Managementtools/_libvirt-Tools/_Die_Managementtools).

```
Host ~ $ sudo vmbuilder kvm ubuntu - suite hardy-o \
- Arch 'i386' \
- Mem '512' \
- Rootsize '8192' - '1024 swapsize' \
- Kernel-flavor 'server' \
- Mirror 'http://archive.ubuntu.com/ubuntu' \
- Components 'main, universe' \
- Addpkg 'openssh-server' \
- Addpkg wget \
- Hostname 'openQRM-test' \
- Name 'openQRM-test' \
- User 'openQRM' \
- Password 'secret'
```

The directory is created to rename and to boot the virtual machine with KVM or QEMU. Access to SSH and HTTP made possible with port redirects.

```
Host ~ $ mv ubuntu-kvm / openQRM-test
Host $ cd ~ openQRM-test
Host ~ $ kvm-m 512-drive file = disk0.qcow2 \
-Net nic-net user, hostfwd = tcp:: 12345 -: 22, hostfwd = tcp:: 8080 - 80
```

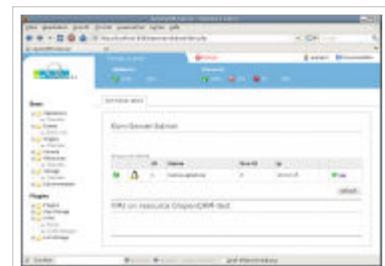
It logs in to the host system.

```
Host ~ $ ssh-p 12 345 openQRM @ localhost
Password: secret
Guest ~ $
```

The package *openQRM-server-4.4-ubuntu804.i386.deb* is to download and install it with *dpkg-i*. A subsequent *apt-get install-f* to solve package dependencies and install the missing packages.

```
Guest ~ $ sudo dpkg-i openQRM-server-4.4-ubuntu804.i386.deb
Guest ~ $ sudo apt-get install-f
```

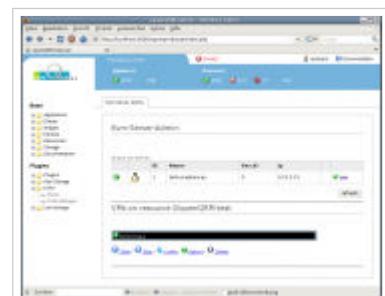
It is to be set after the installation password for the MySQL *root* user asked for the. For the test we used a *secret*. In the file */usr/lib/openQRM/etc/openQRM server.conf* are the MySQL user and password specify which.



openQRM, KVM Plugin - Create new virtual machines, step 2



openQRM, KVM Plugin - Create new virtual machines, step 3



openQRM, KVM Plugin - Create new virtual machine, step 4

```

Guest ~ $ sudo vi /usr/lib/openQRM/etc/openQRM server.conf
OPENQRM_SERVER_BASE_DIR = /usr/lib
OPENQRM_SERVER_INTERFACE = eth0
##### Start of database setup
# Default LAMP setup
OPENQRM_DATABASE_TYPE = "mysql"
OPENQRM_DATABASE_SERVER = "localhost"
OPENQRM_DATABASE_NAME = "openQRM"
OPENQRM_DATABASE_USER = "root"
OPENQRM_DATABASE_PASSWORD = "secret"

```

OpenQRM is then to reboot.

```
Guest ~ $ sudo /etc/init.d/openQRM-server restart
```

The following script, the tables for the MySQL database to be created.

```
Guest ~ $ sudo /etc/init.d/openQRM-server init
```

In addition, log into into the web interface. the web browser is located on the same machine as the virtual machine with openQRM, will log you in with these credentials. The 8080 results are used here port redirect.

```

http://localhost:8080/openqrm/
User: openQRM
Password: openQRM

```

To install the plug-ins URL, the appropriate packages from the <http://sourceforge.net/projects/openqrm/files/> download in a *directory*,*-i* to install it with *dpkg*, and then the package dependencies with *apt-get install-f* dissolve.

```

Guest ~ $ for i in `ls -1 *.deb`, Thu sudo dpkg-i $ i; sudo apt-get install-fy; done
Guest ~ $ sudo apt-get-y autoremove

```

OpenQRM is then to reboot.

```
Guest ~ $ sudo /etc/init.d/openQRM-server restart
```

In the Web Interface *plug-ins* that are installed under - *PlugIn Manager*. The plug-ins are required to activate the *Enabled* button. In this example, the *KVM* ins *LVM storage* and necessary.

Creating virtual machines

The first step is to configure a storage. This is the web interface on the left at the *base* point unfold *Storage* and select *Overview*. The tab is behind *Deployment* of *New Storage Type Lvm Storage Server* to activate. In the *afternoon* *Resource* machine is its own menu. In the new dialog is behind *Storage name*: "testus-storage" to enter. The button ensures you save these settings.

In the second step, an image must be created. This is the web interface on the left at the *base* point unfold *images* and select *Overview*. The tab is in the *New Image Storage* *Storage library* of previously created "testus-storage" option. In the new dialog is behind *name* "testus-image" to enter. The button ensures you save these settings. To check it works with *base* point on the *events* and check the *Events List*.

In the third step, an appliance must be created. This is the web interface on the left at the *base* point unfold *appliances* and select *Overview*. The tab *list* is *New appliances* in the *Resource* own machine "openQRM-test" option. *Name* is behind you "testus-appliances" and turned behind *Resource KVM VM*". The button ensures you save these settings.

In the fourth step, the virtual machine is created. This is the Web interface of the left side of the *plug-in* point and *managers* choose to unfold *KVM KVM*. As a KVM host is activate "testus-appliances". In *Kvm Server Admin* is the green plus sign in the left column of the line "testuslongus to click. In the new dialog *KVM server Create VM* is behind *VM name* "testuslongus" enter. Please click on the button *Create new*

virtual machine to the sets. To check it works with *base* point on the *events* and check the *Events List*.

The new virtual machine is *on VMs resource 0/openQRM-test* listed below. This may take some time. The button renewed *refresh* the display. In the virtual machine can use these buttons *Start, Stop, Config, Reboot* and *Delete* are controlled by the.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_OpenQRM "

This page has been accessed 10,546 times. This page was last updated on 27 September 2010 at 06:22
clock changed. Content is available under GNU Free Documentation License 1.2 .

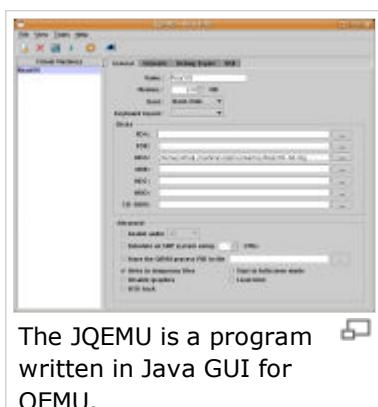
KVM QEMU GUI, Cloud Computing, qemulator, qemu, JQEMU, Qemu Launcher, qemuctl, QEMU Server Tools, Open QEMU Manager, qemubuilder, QWebMon, Libguestfs, xenner convirt, Proxmox Virtual Environment

(Link to this page as [[QEMU-KVM-Buch / Management Tools / More]])

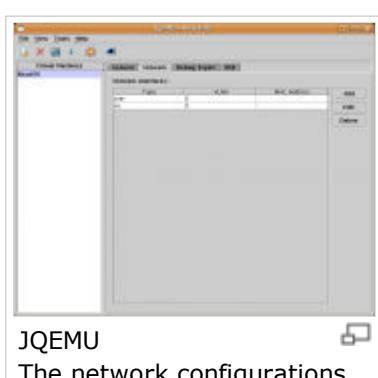
<<< | # # # | >>> | English



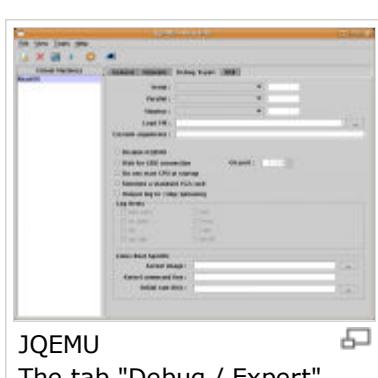
The gemulator.



The JQEMU is a program written in Java GUI for QEMU.



JQEMU
The network configurations.



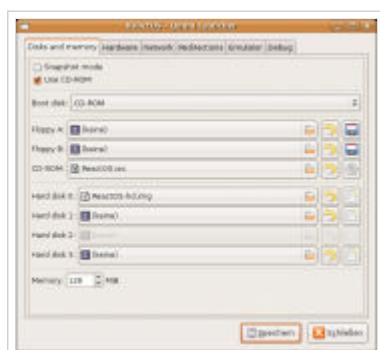
JQEMU
The tab "Debug / Expert".

Contents

- 1 More Tools for QEMU and KVM
 - Archipelago 1.1
 - 2.1 Proxmox Virtual Environment 1.4
 - 1.3 Enomalism
 - 4.1 qemulator
 - 1.5 qtemu
 - 1.6 JQEMU
 - Qemu Launcher 1.7
 - 1.8 qemuctl
 - QEMU Server Tools 1.9
 - 1:10 Open QEMU Manager
 - 1:11 qemubuilder
 - 1:12 QWebMon
 - 1:13 xenner
 - 1:14 convirt



The Qemu Launcher with a virtual machine.



The Qemu Launcher
The VM configuration - Disk and memory.

Other tools for QEMU and KVM

Islands

Website: <http://www.archipelproject.org>

Archipelago is an open source project that aims to bring push notifications to virtualization orchestration using XMPP. Archipelago Relies on open source technologies like Python, ejabberd, cappuccino, disaster, nginx, libvirt, even KVM and more. All of these technologies are mixed together to create a realtime orchestration solution for virtualization.

Proxmox Virtual Environment 1.4

Website: http://pve.proxmox.com/wiki/Main_Page

Proxmox Virtual Environment is a Debian based 64-bit Linux distribution, which serves as a virtualization platform. This distribution includes KVM, OpenVZ and a web interface for managing virtual machines. Will this open source solution installed on multiple machines, so that clusters can be built up (Cloud Computing). Live migration is supported. For instructions on installation can be found at the URL <http://www.howtoforge.com/kvm-and-openvz-virtualization-and-cloud-computing-with-proxmox-ve>.

Enomalism

Website: <http://www.enomalism.com>

Enomalism ECP (Elastic Computing Platform) is a web interface for managing virtual machines on one or multiple hosts (cluster, cloud computing). For instructions on installation can be found at the URL <http://www.howtoforge.com/kvm-virtualization-with-enomalism-2-on-an-ubuntu-8.10-server>.

qemulator

Download: <http://qemulator.createweb.de/de/>

The qemulator is a graphical user interface for QEMU and the Kernel-based Virtual Machine on Linux. On Debian / Ubuntu package is to install *qemulator*.



Qtemu



gemuctl.

```
Host ~ $ sudo apt-get install qemulator
```

For other Linux distributions is the tar file to download and unpack.

```
Host ~ $ tar xzvf qemulator-0.5.tar.gz
Host $ cd ~ qemulator-0.5
```

The directory is part of the graphical installer as administrator (root) starts:

```
Host ~ #. / Setup.py
```

The program is called with *qemulator* or via the *Applications* menu.

```
Host ~ $ qemulator
```

To also virtual machines of the Kernel-based Virtual Machine can manage with the qemulator, KVM is to configure the qemulator. In the menu *settings* in the *Qemu programs for all platforms* we are behind each *name* and *path* "kvm" and then click the *Add* button. Thereafter, the qemulator is to reboot. You choose when configuring a virtual machine under the *Hardware* tab the *system type* "kvm" from.

A new virtual machine I create by clicking on the icon with the plus. A wizard helps you configure. If the virtual machine set up, it will appear in the list. To start choose it from the list and click on the green icon with the triangle. The configuration of the virtual *machine's settings* can be changed the button afterwards. A new image is the menu *system, create new images* created over the.

qtemu

Website: <http://qtemu.org>

Qtemu is another graphical user interface for QEMU. It is based on Trolltech's Qt development environment in the version 4 In Ubuntu package is the installation of a command line does *qtemu*.

```
Host ~ # sudo apt-get install qtemu
```

For FreeBSD and its derivatives, such as PC-BSD, the installation is performed using the following commands.

```
Host ~ # cd /usr/ports/emulators/qtemu
Host ~ # make install clean
```

It is also possible qtemu to install as a package. Package are ready-compiled software packages for FreeBSD. Here are the installation as a package:

```
Host ~ # pkg_add -r qtemu
```

If an appropriate package available to the libraries and tools (*libqt4-dev* *qt4-dev-tools* and *g++*) to install to compile. Thereafter, the source package of qtemu (*qtemu-1.0.4.tar.gz2*) downloaded from the website and unpacked.

```
Host ~ $ tar -x .tar.bz2 xjvf qtemu
```

It is *qtemu-1.0.4* directory to change to that and to generate qtemu.

```
Host $ cd ~ qtemu *
Host ~ $ qmake
Host ~ $ make
```

Qtemu will start the command with *qtemu*.

```
Host ~ $ . / Qtemu
```

To install on Microsoft Windows, the file *qtemu-1.0.5.exe* from the project site to download and run. First, the language for the installation shall be determined. Then the components to install are selected. In the

default setting, all components are enabled. This setting is to take over. The last step is the installation directory for the defined (*C:\ Program Files \ qemu*). The start is *qemu* from the *Start menu*, *qemu*. When you first start you must set the language. This is done via the *File menu*, *Configure* under the *Language* option. In this dialog box is under *MyMachine* the path where the virtual machines to be stored, modified. Also can be adapted to the QEMU start command. Additionally you can define commands to be executed before the start and after you exit QEMU. For the changes to take effect, *qemu* shall cease and re-enter.



It starts the wizard to create a new virtual machine.



It will open an existing virtual machine.



It will start the virtual machine.



It turns off the virtual machine without shutting down.



It is the virtual machine without shutting down starts again.

Table: qemu - the toolbar.

On Linux, *qemu* also be used for the Kernel-based Virtual Machine. This is the *File menu*, *Configure QEMU* after *start command*: "kvm" box. To create a new virtual machine is *new* icon to click on the *machine*. It starts a wizard. First choose an operating system. If desired operating system not listed, choose *Other*. Then the name for the new virtual machine is defined. Depending on the name, the path where the virtual machine is stored, adapted. The next step, the size of the virtual disk is defined. *Qemu* generated images in raw format. Please click on the *Finish* button closes the wizard. The configuration of the new virtual machine is an XML file (. QTE) saved as. You will see a tab with the name of the virtual machine. To switch from one CD / DVD to install the operating system, is under the field of CD-ROM, the path to the CD / DVD image and provide *CD-ROM to boot from* the *Enable* option. The *snapshot mode* option enables *the-snapshot* option. In the area of *cache* memory is the size of the changeable. A floppy *disk* can be integrated in the area. In *network*, it is only possible mode network stack to enable the user. *Others* in the field can be a *transitional option-enable wireless mouse usb tablet device enable*. Furthermore, the number of processors and the system clock can be configured. To start a virtual machine, the icon to click on *Start*.

JQEMU

Download: <http://www.exprofesso.com/jqemu/>

JQEMU is developed in Java on Linux and on platforms that QEMU and Java (JRE / JDK 1.4 +) support, fully functional. Currently dominated JQEMU only the options of the QEMU version 0.8.0. After downloading and unzipping JQEMU is called with the following command.

```
Host ~ $ java-jar jqemu.jar
```

JQEMU manages the QEMU options via GUI and saves the settings in the XML file ~ /. *Jqemu* from. JQEMU needs a few adjustments. *Options* from the *Tools menu* to set the paths to and QEMU *qemu-img* to a. To display errors and other issues can be called JQEMU Console from the *Console* icon or via the *View menu*, to *console*. In the *Tools menu*, or convert *images* generated images. This calls JQEMU tool *qemu-img* to. A new configuration for a virtual machine puts you *add* the icon with. Alternatively, this is achieved via the *File menu*, *Add*. The options are set under the relevant tabs. To save, click on the *Save* icon or *select File*, *Save*.

Under the *General* tab to configure the name of the virtual machine, memory, the boot sequence, the keyboard layout and the paths to the storage media. In the *Advanced* area to enable the sound support and lays down the number of emulated CPUs. There is the possibility to create a PID file. To protect against changes to the images, to enable the protection (*option-snapshot*). The graphics output can be disabled. The full-screen mode can be activated. To install Microsoft Windows 2000 W2K is the option to activate *hack*. DOS-/Windows-Versionen *Local time* for all must be configured.

The tab can *Network* the network settings to adjust. For adding a network setting, click on the *Add* icon. On the tab *Debug / Expert* to debug / experts and special Linux boot options configured. The tab allows the *USB* USB support and you set USB device name. After all settings are made and stored, you start the virtual machine by clicking on the icon *launch*.

Qemu Launcher

Download: <http://download.gna.org/qemulaunch>

The Qemu Launcher manages the QEMU options through a GUI. It should be installed at least version 1.8.0. On Debian and Ubuntu package is the *qemu-launcher* available.

```
Host ~ # apt-get install qemu-launcher
```

For FreeBSD and its derivatives are installing with the following commands.

```
Host ~ # cd /usr/ports/emulators/qemu-launcher
Host ~ # make install clean
```

In many Unix-/Linux-Distributionen, this package is not available or not current. It is then download the program, unpack and install.

```
Host ~ $ tar xzvf qemu-launcher-1.8.0-pre0.tar.gz
Host $ cd qemu-launcher-1.8.0-pre0
```

It is possible to call directly without installing Qemu Launcher.

```
Host ~ $ perl qemu-launcher.pl
```

Qemu Launcher is installed as the system administrator (root).

```
Host ~ # make install
```

After complete installation of the call is as follows:

```
Host ~ $ qemu-launcher
```

The program is configured using the *Settings* icon. Alternatively, this is accomplished via the menu *Tools, Options*. Here is behind the *Path to qemu / usr / bin / kvm* specified, the Kernel-based Virtual Machine use. In *default action* is defined, indicating a name of a virtual machine in the list is to be effected by a double. The appearance of the toolbar with the *Toolbar* icons will be set below. Important setting is under the *Data directory*. This is the directory where the images are typically stored. A new configuration for a virtual machine is created the *new* icon with. Alternatively, this is accomplished using the *Configuration menu, New*. First, after the name of the configuration is required. Then the window to configure the virtual machine is opened.

The first tab defines the QEMU *disk and memory* options for the image and memory. In this example, the options to install ReactOS be applied. It starts from the virtual CD-ROM drive, the image of the installation CD to be indicated. There will be a common virtual hard disk. If such an image does not exist, it is the icon *Create New disk image* created with. The icon is located far right of the line to configure the hard disk. The format *qcow2* is not adjustable. After clicking the button *Save Images* is the name of the new requested. With the icons to the right of the settings for the floppy and the CD-ROM you call a dialog for importing of installation media.

The *Hardware* tab in the Configuration window defines the hardware to be emulated. In this example, apply the default settings. Under the *Network* tab matches a set of network settings to the. In this example, the default settings are applied. The tab *Redirections* changes are not made. The tab *emulator* settings in this example, the default assumed. Even under the last tab *Debug* changes are not made in this example. After clicking on the icons *Save* and *close* the virtual machine appears in the list. By selecting a virtual machine in the list and click on the icon *launch* this virtual machine is started.

qemuctl

Download: <http://qemuctl.sourceforge.net>

qemuctl exists only in version 0.2 and is designed for practical use is not recommended. Furthermore, this version only the QEMU versions up to 0.8.4. qemuctl the instance checks to their maturity. That is, qemuctl is a GUI for the QEMU monitor. With it, you swap from removable media, creates screenshots and virtual machines to freeze (*suspend*). It is only supports the x86 emulation. qemuctl is written in Perl and works as a wrapper script for QEMU. On Debian and Ubuntu installation is done with a command line.

```
Host ~ # apt-get install qemuctl
```

In many distributions this package is not available. It is therefore advisable to download the program from the project website, to unpack and as a system administrator (root) to install. Note here the installation instructions at this URL.

```
Host ~ # tar xvf qemuctl0.2.tar.gz
Host ~ # cd qemuctl0.2
Host ~ # make install
```

In the simplest case, *qemuctl qemu* when starting a virtual machine specified instead. The *options-serial stdio-monitor* and can not be applied with *qemuctl* because they *qemuctl* communication with QEMU uses.

```
Host ~ $ qemuctl Platte.img
```

It appears next to the QEMU window a small window with the qemuctl menu bar. The QEMU monitor is turned off, as its functions are accessible via the menus of qemuctl.

Menu	Function
File	About the <i>File</i> menu allows Screenshots (. <i>Ppm</i>) and include states of virtual machines or secure the load (. <i>Vm</i>). Qemuctl <i>Quit</i> exits and the virtual machine. It is cheaper at the start of qemuctl <i>with-suspend-dir</i> directory for the files provide a Suspend, as these are stored in the directory home otherwise and each other to write. Also, a suspend file with <i>the-suspend-file</i> be loaded at startup to saved state the virtual machine to load the inside. The file is no extension. Specify <i>vm</i> .
Connect	Access to removable media, USB, serial and parallel ports, you can control the functions in the menu on <i>Connect</i> .
Macros	<i>Macros</i> menu are managed. It is the macro <i>Suspend</i> available. It stops the emulation and submit a screenshot (. <i>Ppm</i>). Thereafter, the state of the virtual machine in a file (. <i>Vm</i>) secured and QEMU ended.
Signal	The menu commands <i>signals</i> send signals to the virtual machine. The menu item sent <i>Send Keys</i> Keys for instance. Furthermore, it freezes there a virtual machine, or can continue to run it again. In addition, we switched here from the virtual machine, or restarts. With <i>Commit</i> save you changes in the <i>snapshot</i> mode.
Debug	About the <i>Debug menu</i> , <i>Shell</i> is a window with a shell for QEMU monitor commands open.
Help	The help menu, <i>Help</i> is not implemented. Help you get the man pages.

QEMU Server Tools

Download: <http://qemu-server.sourceforge.net>

The QEMU Server Tools were developed with the aim to manage multiple instances running permanently on a Linux host. For each virtual machine its own directory is created. In it are next to one or more image files, other files to control and configure the virtual machine. Sorry, no clean shutdown of the guest systems is implemented, as for example in VMware ESX Server with VMware Tools possible. This is necessary for a shutdown of the host. For host systems that do not have to store the data (for example, printer server, or honeypots) would be *option-snapshot* a way to circumvent this problem. The instances running in the background and a VNC remote access is enabled. The inputs to the QEMU monitor via a named pipe. Named Pipes are used for communication between processes that are not related and, moreover, may reside on different computers within a network. The expenses of the QEMU monitor can be saved to a log file. The QEMU written in Perl Server tools are released under the GNU General Public

License. To install QEMU be provided (version 0.8.1) and Perl (version 5.8.x). You are advised to KQEMU (version 1.3.0pre9 from) and the *bridge-utils*. For a live performance is still a slim Linux distribution to recommend, for example, Ubuntu Server or Debian. After downloading, unpack the archive and change to the directory created.

```
Host ~ # tar xzvf qemu-server-0.2a.tar.gz
Host ~ # cd qemu-server
```

Installation is done with two commands.

```
Host ~ # make
Host ~ # make install
```

The call of the QEMU Server Tools should not be the *root* user under. Therefore, the following directories should be located and then transfer them to the user under which the QEMU Server Tools should be started. The directory */var/run/qemu* is to store the PID file.

```
Host ~ # mkdir / var / run / qemu
Host ~ # chown USER: GROUP / var / run / qemu
```

In the */vm* machines are the directories for virtual created. This directory should be for a live performance on a separate partition.

```
Host ~ # mkdir / vm
Host ~ # chown USER: GROUP / vm
```

The tool *qemu-server-createvm* helps to create a virtual machine. It creates a directory with two files. First, it uses *qemu-img* to image (*disk1.raw*) in raw format to generate a disk. Second, this is a configuration file (*qemu.conf*) written. The general syntax is:

```
Host ~ $ qemu-server-createvm [vmdir] [size]
```

vmdir defines the path of the directory and *size* sets the size of the virtual disk. Here is an example of ReactOS:

```
Host ~ $ qemu-server-createvm / vm / reactos / 1G
```

The directory of the virtual machine from the QEMU Server Tools *qemu.conf* requires a configuration file. This file must be in accordance with the desired QEMU options to be adjusted. Each line in this file corresponds to a QEMU-boot option. Here is an example of ReactOS:

```
hda = ReactOS.img
boot = c
net = nic
net = user
kernel-kqemu
local time
snapshot
vnc = localhost: 1
```

To start the instance script is the *Perl-qemu-server* with the *start* option and the path to the virtual machine. The use is similar to *vmrun* VMware Server.

```
Host ~ $ qemu-server start / vm / reactos /
```

Since the instance is running in the background and VNC is a remote access to the desktop of the virtual machine is not much to see. An overview of the current instance, the tool *qemu-server* with the option *list*.

```
Host ~ $ qemu-server list
vmdir vnc pid
-----
```

```
/ Vm / ReactOS / 0 21 727
Total running VMs: 1
```

This example is an instance. It can create and launch more instances. It connects to a VNC viewer to the virtual machine.

```
Host ~ $ vncviewer localhost: 1
```

The virtual machine will have the option to *stop*.

```
Host ~ $ qemu-server stop / vm / reactos /
```

The tool *qemu-server* continues to *reset*, *pause*, *resume* and *suspend* callable.

```
Host ~ $ qemu-server reset / vm / reactos /
Host ~ $ qemu-server pause / vm / reactos /
Host ~ $ qemu-server resume / vm / reactos /
~ $ Qemu-server host suspend / vm / reactos /
```

The following command saves the state of the virtual machine are located in *qemu.state* (up QEMU 0.8.2). Requirement is the *option-snapshot*.

```
Host ~ $ qemu-server commit / vm / reactos /
```

The following command loads the state from the file *qemu.state* the virtual machine (QEMU to 0.8.2).

```
Host ~ $ qemu-server restore / vm / reactos /
```

The following command runs the QEMU monitor command *info* from *network*.

```
Host ~ $ qemu-server exec / vm / reactos / 'info network'
```

Open QEMU Manager

Download: <http://sourceforge.net/projects/openqemumanager/>

Open QEMU Manager is a web application to manage virtual machines.

```
Host ~ $ tar xzvf openqemumanager-0.4.tar.gz
Host $ cd ~ openqemumanager
Host ~ $ less README.txt
```

qemubuilder

Website: <http://wiki.debian.org/qemubuilder>

QWebMon

Website: http://research.xlab.si/index.php?option=com_content&task=view&id=107&Itemid=142

QwebMon is a web application to manage virtual machines.

```
Host ~ $ wget http://research.xlab.si/download/qwebmon-0.1.0.tar.gz
Host ~ $ tar xzvf qwebmon-0.1.0.tar.gz
Host $ cd ~ qwebmon-0.1.0
Host ~ $ less README
```

xenner

Website: <http://kraxel.fedorapeople.org/xenner/>

Xen guest machines can be operated under the Kernel-based Virtual Machine use the tool xenner.

convirt

Website: <http://www.convirt.net>

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Managementtools/_Weitere "

This page has been accessed 9456 times. This page was last updated on 6 May 2010 at 17:21 clock changed. Content is available under GNU Free Documentation License 1.2 .

QEMU QEMU KQEMU Support Download 0.11.1, invalid option-kernel-kqemu

(Link to this page as [[QEMU-KVM-Buch / QEMU + KQEMU on Linux]])

<<< | # # # | >>> | English

Contents

1 KQEMU QEMU and Linux
1.1 Software packages
1.2 Source compile
1.2.1 Ubuntu 9.10 and 10.04
1.2.1.1 KQEMU 1.4.0 pre1
1.2.1.2 QEMU 0.11.1
1.2.2 Fedora 14
1.3 Special commands and options for KQEMU
4.1 Quick Start

QEMU and Linux KQEMU

The support for the QEMU accelerator KQEMU ends with 0.11. Under Ubuntu 9.10 (GNOME) package provides the QEMU *qemu-kvm* with 0.11.0 KQEMU not already support. From QEMU 0.12.0, only the Kernel-based Virtual Machine (KVM) is supported. On computers without hardware virtualization technology is KVM but not used. Who would not buy any new hardware and wants to accelerate QEMU still need to install QEMU 12.11.* or older. In parallel, you can also use a newer version of QEMU KQEMU without support. For example, installed the Ubuntu package the QEMU binaries in */usr/bin/*. The compiled version is under */usr/local/bin/*. Specifying the full path in each case, the version you wish to call.

Software packages

When you install QEMU with software packages can support the KQEMU not be guaranteed. Even if the current package still offers KQEMU support, this may be different for an update. recommend, therefore, is compiling QEMU. KQEMU can be installed as a package. It should be noted that the packages will be obsolete in a newer version of the distribution can. For Red Hat and Fedora to Load the RPM packages from the URL <http://atrpms.net/name/kqemu/> down. Around the accelerator KQEMU to install Ubuntu on, packets are the *kqemu-common*, *kqemu-source* and *module-assistant* to install.

```
Host ~ $ sudo apt-get install kqemu-common kqemu-source
```

In versions prior to Ubuntu 9.10 the following steps are necessary.

```
Host ~ $ sudo apt-get install module-assistant
Host ~ $ sudo module-assistant prepare
Host ~ $ sudo module-assistant auto-install kqemu
```

The *lsmod* command shows whether the module was successfully loaded by the operating system *kqemu*.

```
Host ~ $ lsmod | grep kqemu
kqemu 123 940 0
```

compiling sources

Ubuntu 9.10 and 10.04

KQEMU 1.4.0 pre1

The sources of KQEMU are to compile as usual. Before packages are *gcc*, *make* and install *wget*. On Debian / Ubuntu package, the tools to compile the combined *build-essential*.

```
Host ~ $ sudo apt-get install make gcc build-essential wget
```

recommend to *check* the package continues to *install*.

```
Host ~ $ sudo apt-get install checkinstall
```

The sources are downloaded, unpacked and compiled.

```
Host ~ $ mkdir-p ~ / source
Host $ cd ~ / source
Host ~ $ wget http://qemu-buch.de/download/kqemu-1.4.0pre1.tar.gz
Host ~ $ tar xzvf kqemu-1.4.0pre1.tar.gz
Host $ cd kqemu-1.4.0pre1
Host ~ $. / Configure
Host ~ $ make
```

recommend software package is to generate a *check* to *install*. It is called instead of *make install*. A support can be obtained with *checkinstall - help*. The program provides *checkinstall* some questions that the specifications can be answered with mostly default. Clearly, however, the package name to specify, for example *kqemu-self-compiled*. In the current directory the package is created and installed. On Debian / Ubuntu it lists the command *dpkg-l* and *dpkg-r* uninstall it if necessary.

```
Host ~ $ sudo checkinstall - fstrans = no - pkgname = kqemu-self-compiled
```

You can alternatively install the software in a conventional manner.

```
Host ~ $ sudo make install
```

The compiled kernel module is loaded with the *modprobe* command.

```
Host ~ $ sudo modprobe kqemu
```

This is the device / *dev / kqemu*.

```
Host ~ $ ls-1 / dev / kqemu
crw-rw-rw-1 root root 250, 0 2009-03-17 14:24 / dev / kqemu
```

On that device with virtual machine communicate with the kernel module *kqemu*. To test QEMU is the *option-kernel-kqemu and-monitor stdio* called with. It may receive the error message *qemu: invalid option* does not appear. With the QEMU monitor command *info kqemu* you check the support of KQEMU.

```
Host ~ $ qemu-kernel-kqemu-monitor stdio
(Qemu) info kqemu
kqemu support: enabled for user and kernel code
```

Sometimes when you start QEMU output this error message:

```
Could not open '/ dev / kqemu' - QEMU acceleration layer not activated
```

Then the following commands are run:

```
Host ~ $ sudo mknod / dev / kqemu c 250 0
Host ~ $ sudo chmod 666 / dev / kqemu
```

Thus, the module at startup of the host is automatically loaded, Debian derivatives a line with *kqemu* in / *etc / modules* must be entered. That the access rights of // *kqemu* also restart properly be set according to the *dev*, is the command *chmod 666 / dev / kqemu* the file / *etc / rc.local* entered into. If / *dev / kqemu* even after the computer has been restarted, the installation was successful.

Newer Linux distributions with kernel 2.6 instead of the elderly use the new *udev* *devfs* to manage the kernel device files. This makes it possible hotplug devices at run time to integrate automatically. The configuration of *udev* is using configuration files in / *etc / udev*. If the Linux distribution supports *udev* is the device / *dev / kqemu* not automatically created when booting. These are in a boot script as / *etc / rc.local* the following lines to write.

```
mknod / dev / kqemu c 250 0
chmod 666 / dev / kqemu
```

QEMU 0.11.1

0.11.1 to compile QEMU these packages are necessary:

```
Host ~ $ sudo apt-get install make gcc wget
Host ~ $ sudo apt-get install zlib1g-dev libsdl-dev gfax1.2
```

recommend the package *checkinstall*.

```
Host ~ $ sudo apt-get install checkinstall
```

The following packages are not strictly necessary but can extend the functionality of QEMU. For the security of VNC sessions are the packages *devel cyrus-sasl-devel* and *gnutls-install*. The VDE VDE Library creating virtual switches is possible.

```
Host ~ $ sudo apt-get install libsasl2-dev libgnutls-dev
Host ~ $ sudo apt-get install libvdeplug2-dev
```

The library *curl-devel* supports booting via web protocols.

```
Host ~ $ sudo apt-get install gnutls-dev-libcurl4
```

For Bluetooth support packages will and *bluez-dev libbluetooth* needed.

```
Host ~ $ sudo apt-get install bluez-dev libbluetooth
```

Support for Xen, the compiler expects the library *libxen3-dev*.

```
Host ~ $ sudo apt-get install libxen3-dev
```

To generate the documentation *qemu-doc.html* package is the *texi2html*.

```
Host ~ $ sudo apt-get install texi2html
```

If the sparse checker (*CGCC*) is needed, the package is to install *sparse*.

```
Host ~ $ sudo apt-get install sparse
```

The *library-dev libbrlapi* is used for the braille edition.

```
Host ~ $ sudo apt-get install libbrlapi-dev
```

First, the source packages for QEMU be downloaded and unpacked. Compile it with normal user rights *with. / Configure & & make*. Initial installation requires root privileges. With *checkinstall* or *make install* the files created in the */usr / local / bin* and */usr / local / share / qemu* copies.

```
Host ~ $ mkdir-p ~ / source host ~ $ cd ~ / source host ~ $ wget \ http://download.savannah.gnu.org/releases/qemu/qemu-0.11.1.tar.gz
```

recommend software package is to generate a *check* to *install*. Clearly, the package name is to specify, for example *qemu-kqemu-support-with-self-compiled*.

```
Host ~ $ sudo checkinstall - fstrans = no - pkgname = qemu-self-compiled
```

You can alternatively install the software in a conventional manner.

```
Host ~ $ sudo make install
```

A look at the file *qemu-doc.html* is recommended. Ubuntu QEMU package was previously installed, this is the path */usr / bin /*. The compiled version is under */usr / local / bin /*. Specifying the full path in each case the version is called.

```
Host ~ $ / usr / bin / qemu
QEMU PC emulator version 0.13.0, Copyright (c), Fabrice Bellard
Host ~ $ / usr / local / bin / qemu-version
QEMU PC emulator version 0.11.1, Copyright (c) 2003-2009 Fabrice Bellard
```

Fedor a 14

To compile these packages are installed:

```
Host ~ $ su -
Host ~ # yum install wget gcc-devel texi2html zzip lib
Host ~ # yum install kernel-headers kernel-devel
```

```
Host ~ $ exit
```

The sources of KQEMU be downloaded, unpacked and compiled.

```
Host ~ $ mkdir-p ~ / source
Host $ cd ~ / source
Host ~ $ wget http://qemu-buch.de/download/kqemu-1.4.0pre1.tar.gz
Host ~ $ tar xzvf kqemu-1.4.0pre1.tar.gz
Host $ cd kqemu-1.4.0pre1
Host ~ $ . / Configure & & make
```

The installation is the user under *root*. Then the compiled kernel module is loaded with the *modprobe* command.

```
Host ~ # make install
Host ~ # modprobe kqemu
```

In the file */etc/rc.local* the following lines are indicated:

```
modprobe kqemu
chmod 666 / dev / kqemu
```

The sources of QEMU 0.11.1 be downloaded, unpacked and compiled.

```
Host $ cd ~ / source host ~ $ wget \ http://download.savannah.gnu.org/releases-noredirect/qemu/qemu-0.11.1.tar.gz
```

The installation is the user under *root*.

```
Host ~ # make install
```

Specific options and commands for KQEMU

If the host and guest system available in x86 processor architecture, QEMU can use with the optional accelerator KQEMU the Native Virtualization. It goes KQEMU most commands directly to the real CPU. Only the CPU commands that directly address the hardware are intercepted and replaced with custom routines. For Linux guest systems the best Acceleration with the 2.4 kernel is reached. The 2.6 versions work, but are slower. To enable the full KQEMU virtualization is the *option-kernel-kqemu* apply. The QEMU monitor shows *info kqemu* command the use of the KQEMU.

```
Host ~ $ qemu-monitor stdio-kernel-kqemu
(Qemu) info kqemu
kqemu support: enabled for user and kernel code
```

Another *option-enable-kqemu*. Only KQEMU is enabled in the user code.

```
Host ~ $ qemu-monitor stdio-enable-kqemu (qemu) info kqemu kqemu support: enabled for user code
```

So you do not always command *qemu-kernel-kqemu* enter the must submit an alias in *~/.Bash_aliases* (Ubuntu), and *~/.Bashrc* (Fedora) to. One need then only enter *kqemu*.

```
aka kqemu = "qemu-kernel-kqemu"
```

If the use is to be prevented from KQEMU is in older versions of *QEMU-the-kqemu* indicate *no*. When you install some guest operating systems, such as, for example, Microsoft Windows, KQEMU not apply. That is, when the deactivation of the KVM hardware virtualization *kvm-no* recommended *is-is* here KQEMU disable. After installation, KQEMU be enabled on all versions of Windows.

```
Host ~ $ qemu-monitor stdio-no-kqemu
(Qemu) info kqemu
kqemu support: disabled
```

In this case KQEMU has been disabled. In the example below KQEMU QEMU compiled without support. *The-kernel-kqemu-no-kqemu* and are not available.

```
(Qemu) info kqemu
kqemu support: not compiled
```

Quick Start

A Guide to Getting Started with QEMU can be found at the URL <http://qemu-buch.de/d/Quickstart> .

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_QEMU%_2BKQEMU_unter_Linux "

This page has been accessed 10,218 times. This page was last updated on 25 January 2011 at 07:53 clock changed. Content is available under GNU Free Documentation License 1.2 .

The version without GUI is suitable to burn a DVD with a virtual machine and the QEMU software. The virtual machine on this DVD can be started on another computer without installing QEMU.

QEMU and KQEMU

Download:
<http://download.superjobmagazin.com/install/>
[http://wiki.qemu.org/Download_\(KQEMU\)](http://wiki.qemu.org/Download_(KQEMU))
<http://www.qemu.org/index.html>

To install QEMU without a GUI and without KQEMU no administrator rights are required and it is not the Windows Registry changes. First, the latest version is downloaded and unpacked:

QEMU supports up to version 0.11.1 the optional accelerator KQEMU. To install KQEMU file is first to unpack the J.1.Qpret.tar.gz file. Right-click on the file Kqemu.nrf With the installation begins. Then, the computer is restarted. After that the command can be entered:

```
root:~> net start qemu
```

With net start on Windows service started. This line should be inserted in a startup script so that they will not have to type after every reboot. Quick Start section in the application of a virtual machine is the command line described (see <http://qemu-buch.de/d/Quickstart>). It must first change to the directory with the QEMU binary. It is also possible to include this directory in the PATH variable. The PATH variable is Microsoft Windows XP in the Control Panel under System properties, Advanced, Environment variables defined. A comfortable console supported on Cygwin or WindowsPowerShell (see http://qemu-buch.de/d/Anhang/_Nutzliche_Tools).

Win32-cross-compiler
Download: <http://sourceforge.net/projects/qemu/>
Win32QEMU is a native Windows version of QEMU. The source code of QEMU for Visual C + + 2008 ported. The installation of Win32QEMU is simple, does not modify the Windows registry and they need not have administrative privileges. To install the latest version of Win32QEMU is downloaded and unpacked. In the DOS command prompt, change to the directory created. Instead qemu.exe Win32QEMU/test.exe to call.

compiling sources

Cross-compiling for Linux

Linux brings not only the necessary tools to compile it, it also supports compiling for other architectures. In this example, the generation of the Windows binaries under Linux distribution Fedora 14m. In addition to the required packages to compile (see <http://qemu-buch.de/d/QInstallation>) Windows is a cross-compiler necessary. This is the package mingw32-gcc posted about. Furthermore, the MinGW versions of the SDK and the zlib library is required.

```
host:~> % wget -c "http://www.mingw.org/distros/mingw32-gcc/mingw32-zlib-0.9.2.patch" wine"
```

As newer versions of QEMU support mostly with patches of the source code is available (see <http://distcc.org/windows/qemu/>), in this example, the cross-compilation for QEMU 0.11.0. First, the archive containing the source code downloaded and unpacked in a new directory.

```
host:~> % cd /source/host ; tar -xvf /http://download.savannah.gnu.org/releases/qemu/qemu-0.11.1.tar.gz host -D tar xvzf qemu-host-0.11.1.tar.gz & cd qemu-0.11.1
```

The path variable PATH is part of the MinGW tools to expand the.

```
host:~> % export PATH=/usr/1485-pc-mingw32/sys-root/mingw/bin:$PATH
```

The configuration of the compiler is run as usual, configure with the command, here the cross-compiler and defines the installation path. Then start using make to compile.

```
host:~> % ./configure --cross-prefix=i686-pc-mingw32- --prefix=/tmp/qemusinstall /
```

host:~> % make

The installation path is created.

```
host:~> % cd /tmp/qemusinstall /
```

Make install the files necessary to the installation path copied.

```
host:~> % make install
```

Furthermore, manually copy these files:

```
host:~> % cp ./i686-pc-mingw32/include/* /sys-root/mingw/include/
```

```
host:~> % cp ./i686-pc-mingw32/lib/* /sys-root/mingw/lib/
```

Wine binaries with the first test subject.

```
host:~> % ./mingwexec0
```

```
host:~> % ./mingwexec0
```

```
host:~> % ./mingwexec0
```

The files in this folder are copied to a Windows machine in a directory.

MobaLiveCD

Download: <http://mobalived.mobalived.net/>

A to boot from live CDs and USB sticks used restricted version of QEMU MobaLiveCD. This tool will be downloaded to any directory and start zuarbeiten zu ISO images of live CDs. Comfortable, booting the ISO image by right-clicking the ISO file. This menu is before this function with the button right-click to activate. QEMU launch options can not pretend to. The QEMU monitor is used.

Tools

To use all possibilities of the Windows version of QEMU is to install additional tools. Other useful tools are described in the Appendix (see http://qemu-buch.de/d/Anhang/_Nutzliche_Tools).

/ TUN/TAP devices

Microsoft Windows do not offer TUN/TAP support. This can only be established by the installation of the TUN/TAP adapter OpenVPN. In the delivery of Qemu-Hammer is the 32-bit version of TUN/TAP adapter (C:\Program Files\Qemu-Hammer\TAPDriver\1.0\OpenVPN\1.0\OpenVPN\1.0.2.20.exe). To install the original TUN/TAP adapter OpenVPN Windows installation file from Microsoft: <http://openvpn.net/index.php/open-source/downloads.html>

```
host:~> % openvpn-show-adapters
```

Available TAP/WIN32 adapters [name, GUID]:

```
host:~> % openvpn-show-adapters
```

Available TAP/WIN32 adapters [name, GUID]:

```
host:~> % openvpn-show-adapters
```

In this example 2, an adapter name Local Area Connection created. In the Control Panel gives you this one simple name, such as my-tap. The success of the renaming of tabs following command:

```
host:~> % openvpn-show-adapters
```

Available TAP/WIN32 adapters [name, GUID]:

```
host:~> % openvpn-show-adapters
```

It is assigned to this interface an IP address. This is in the Control Panel Network Connections select the adapter. Under the item Internet Protocol (TCP / IP) is the button to click Properties. The IP address must be on the same network segment as the host of the system are and must not already assigned.

```
host:~> % openvpn-show-adapters
```

When calling QEMU with these options, the network connection my-tap the specified IP address (10.0.2.20) is assigned (see http://qemu-buch.de/d/Netzwerkoptionen/_Virtuelle_Netze_konfigurieren). Some host systems may need the network configuration from DHCP to a static IP address to be changed, for example, 10.0.2.15. If a Windows Firewall active, this for the TAP adapter is disconnected.

Ibususb-win32

Download: <http://sourceforge.net/projects/ibususb-win32/>

The USB support for Microsoft Windows is still in an experimental phase and supports USB hard disk images, virtual USB mice and USB virtual keyboards. Access to the USB devices on the host system slows from QEMU. It is the library ibususb-win32 necessary. This has been ported from Linux to Microsoft Windows. To install the Library invites you to the file ibususb-win32-filter-bin-0.1.22.1.exe download and start it. As usual on Windows, is to click Run several times. With the included test program to determine the product name, the vendor ID and product ID of the USB devices on the host. In Qemu Manager under the Hardware tab to enable the USB support, and with information obtained (product name, vendor ID and product ID) as a host device to the USB devices one. The QEMU monitor can be reviewed with info USB host availability of the host device.

```
(ibus) info USB host
```

```
Device 0:14: Apen 400 MB / Class 00: USB device 980c: 1000, USB 2.0
```

Creating virtual machines

The creation of virtual machines at the URL <http://qemu-buch.de/d/Gast-Systeme>.

QEMU on earlier versions of Microsoft Windows

On Microsoft Windows 2000 SP4 QEMU 0.1.1 or QEMU to 0.11.0. Under older versions of Microsoft Windows QEMU older versions are used.

```
host:~> % cd /source/host /qemu-0.11.0
```

```
host:~> % http://qemu-buch.de/download/qemu-0.15.1-windows.zip
```

```
host:~> % http://qemu-buch.de/download/qemu-0.15.2-windows.zip
```

QEMU under Wine

Wine (Windows Emulator) is a Windows-compatible runtime environment for POSIX-compatible operating systems such as Linux. This makes it possible for Microsoft Windows operating systems, programs compiled under BSD Unix (*BSD, Solaris, Mac OS X) to run Linux (see http://qemu-buch.de/d/Anhang/_Nutzliche_Tools/_Wine). Also Windows version of QEMU can run under Wine without KQEMU. Requires the host in QEMU Devices Manager. The QEMU Manager is a virtualizer. As KQEMU is not an emulator it is a virtualizer still, the execution speed of QEMU under Wine is hardly less than QEMU without KQEMU under Microsoft Windows. However, the graphics output on Microsoft Windows is usually faster. QEMU is useful to include tests under Wine. Is compiled for QEMU for Windows on Linux with Wine, the result must be treated first. Continue to use the Windows Client ReactOS (http://reactos.org) source code from Wine.

The QEMU Manager for Windows 6.0 is installed under Wine with the same steps as in Microsoft Windows. The installation of KQEMU must be disabled here. When QEMU version without GUI, only the archive with the QEMU binaries downloaded into a directory under ~/wine/drive_c/ unpacked, QEMU can be run in Wine console.

```
host:~> % cd /source/host /qemu-0.11.0
```

```
host:~> % http://qemu-buch.de/download/qemu-0.15.1-windows.zip
```

```
host:~> % http://qemu-buch.de/download/qemu-0.15.2-windows.zip
```

```
host:~> % ./qemu-0.11.0 -H wine
```

QEMU can also be accessed directly in the Linux shell.

```
host:~> % wine c:\qemu\qemu.exe -L c:\qemu-hda c:\qemu\linux.img
```

The 64-bit version starts with the following command.

```
host:~> % wine c:\qemu\qemu-system-x86_64.exe -L c:\qemu-hda c:\qemu\linux.img
```

For problems file, the error messages in the written stderr.txt.

```
host:~> % cat ~ /wine /drive_c /qemu /stderr.txt
```

A simplified form of the call of the QEMU binaries under Wine is available by defining aliases. This works for example in ~/Bashrc mentioned as follows. After re-login there are the aliases available.

```
alias wqemu='wine $HOME/.wine/drive_c/qemu/' alias wqemu-system-x86_64='wine $HOME/.wine/drive_c/qemu/qemu-system-x86_64.exe' alias wqemu-img='wine $HOME/.wine/drive_c/qemu/qemu-img'
```

It is 32-bit emulator wqemu and the 64-bit emulator wqemu-system-x86_64 started the.

```
wine & http://qemu-buch.de/d/Gast-Systeme
```

Retrieved from http://qemu-buch.de/d/index.php/QEMU-KVM-Buch/_QEMU_Unter_Microsoft_Windows

This page has been accessed 31,899 times. This page was last updated on 23 January 2011 at 19:40 clock changed. Content is available under GNU Free Documentation License 1.2.

QEMU 0.11.0 OS / 2 port - QEMU for OS / 2 Warp 4 and eComStation 2.0 - Virtualization and Emulation

(Link to this page as [[QEMU-KVM-Buch / QEMU under eComstation]])

<<< | # # # | >>> | English

Contents

- 1 QEMU for OS / 2 Warp 4 and eComStation 2.0
 - 1.1 QEMU 0.11.0 OS / 2 port
 - 2.1 Quick Start
 - 1.3 Additional options of QEMU OS / 2 ports

QEMU for OS / 2 Warp 4 and eComStation 2.0

QEMU 0.11.0 OS / 2 port

Download: <http://www.os2site.com/sw/emulators/qemu/index.html>

This port for OS / 2 Warp 4 and eComStation is in development and still experimental. Not yet implemented include the accelerator KQEMU, the emulation of other processor architectures and the sound output. Also, management tools have not been ported. The configuration of the virtual machines should be issued from the command line.

System requirements are at least a Pentium CPU, OS / 2 V4.50 (Warp V4 + FP13, WSeB, MCP, ACP), or eComStation, and the library TCPIP32 klibc 0.6.3. It recommends that the *CONFIG.SYS* file to the line *CLOCK SCALE = 4* is added. In Warp4/WSeB an update of the kernel is necessary. Scitech SNAP is a favorable or GRADD-based graphics driver for SDL.

An installation of QEMU is not necessary. It will only download the archive with the latest version of QEMU and unpacked. That is all. It can also be different versions of QEMU running simultaneously. In *command prompt*, change to the created directory and start *qemu.exe*.

```
Host [C: \] qemu-version
SYS1804: The system can not find the file LIBC063.
```

This error message is that the library from the URL LIBC063 [ftp://ftp.netlabs.org/pub/gcc/libc-0_6_3-csd3.wpi](http://ftp.netlabs.org/pub/gcc/libc-0_6_3-csd3.wpi) download and double click to install one.

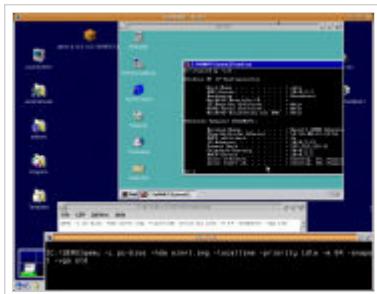
```
Host [C: \] qemu-version
SYS1804: The system can not find the file KSDL12.
```

This error message appears when the library is missing KSDL12.dll on the system. This is according to Hobbes, the appropriate archive zusuchen: <http://hobbes.nmsu.edu/h-search.php?button=Search&key=SDL-1.2.14-os2-gcc44-20091106>. After downloading the file archive is unpacked and *KSDL12.dll* from the dll path to copy the. After successful installation shows *qemu-version* and the version of QEMU *qemu-help* options to the.

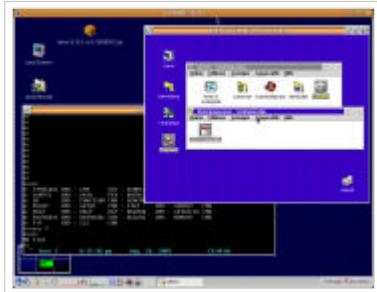
```
Host [C: \] qemu-version
QEMU PC emulator version 0.11.0, Copyright (c) 2003-2008 Fabrice Bellard
Host [C: \] qemu-help
...
```

Quick Start

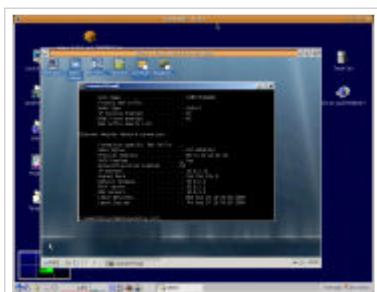
For the establishment of a virtual machine at least one image file and an operating system is required as a guest. With the QEMU virtual machine is configured options. When installing the guest systems is to support the hardware emulated by QEMU



Microsoft Windows NT 4.0
as a QEMU guest systems
under eComstation.



CP/M-86 and OS / 2 Warp 4
as a QEMU guest systems
under eComstation.



ReactOS as a QEMU guest system under eComstation.
The network configuration of the host system is provided by the built-in DHCP Server QEMU.



ReactOS in the 32 - and 64-bit emulation QEMU.

respected.

For this example (which was the operating system ReactOS <http://www.reactos.org/de/>) selected as the guest. For the following reasons:

- The installation complies with the Microsoft Windows systems and is therefore well known.
- ReactOS is Open Source and freely available.
- The image of the installation CD is downloaded quickly.

To install the operating system a virtual disk is created. This is done with the package contained in the QEMU *qemu-img* tool and its parameters *create* (see also http://qemu-buch.de/d/Speichermedien/_Images_anlegen). Given the name and size of the virtual disk to be indicated. With *the-f* format, the set (see http://qemu-buch.de/d/Speichermedien/_Image-Formate).

```
Host [C: \] qemu-img create-f qcows ReactOS.img 1G
```

From the ReactOS website is the image of the installation CD to download and unpack.

```
Host [C: \] unzip ReactOS *- iso.zip
```

Is called QEMU *qemu* with the command. The configuration of the virtual machine is in options on the command line. To install the host system, the virtual machine is started with the image of the installation CD. Whose name is entered after the *option-cdrom*. The name of the created virtual hard disk is *hda-entered* (see also on the option http://qemu-buch.de/d/Speichermedien/_Zugriff_auf_Speichermedien). This virtual computer from the CD will boot the virtual, *d* is the *option-boot* responsible. *The-L* refers to the directory containing the BIOS files. When QEMU for OS / 2 and eComStation is also the *option* to use *priority* (see below).

```
Host [C: \]-L qemu pc-bios-hda-cdrom ReactOS.img ReactOS.iso-boot d-priority idle
```

It will start the virtual PC in a window. Installing ReactOS is started by clicking in the window and pressing any key. First, the German keyboard is selected. Thereafter, the virtual disk is partitioned and formatted. The target directory for the installation of the ReactOS files need not be changed. After the boot loader is installed, the default option *Install bootloader on the harddisk (MBR)* holds. Then the virtual machine must be restarted. The installation continues graphically. To the hard drive to boot from, *c* is the *option-boot* to enter. The QEMU window is left [Alt] [Ctrl] +.

```
Host [C: \]-L qemu pc-bios-hda-cdrom ReactOS.img ReactOS.iso-boot c-priority idle
```

To boot without the CD image, the command applies.

```
Host [C: \]-L qemu pc-bios-hda ReactOS.img-local time-priority idle
```

It is recommended to install and configure the virtual machine shut down and one or more overlay to create files to image to make changes to protect the base (see # http://qemu-buch.de/d/Speichermedien/_Images_anlegen Overlay Images_anlegen). In this example, two overlay files can be created.

```
Host [C: \] qemu-img create-f-b ReactOS.img qcows ReactOS 01.ovl
Host [C: \] qemu-img create-f-b ReactOS.img qcows ReactOS 02.ovl
```

It is now possible to connect two QEMU instances, each with an overlay file to start. The base image with the default installation remains unchanged.

```
Host [C: \]-L qemu pc-bios-hda ReactOS 01.ovl-local time-priority idle
Host [C: \]-L qemu pc-bios-hda ReactOS 02.ovl-local time-priority idle
```

ReactOS to 64-bit processor architecture to test under is, *qemu-system-x86_64* apply.

```
Host [C: \] qemu-system-x86_64-pc-bios-hda L ReactOS 01.ovl-local time-priority idle
```

So you do not always command line to start a virtual machine must complete the type, file, the command line in a with *the*. Written *cmd*. This file is clicked, starts the virtual machine. For manipulating the running virtual machine is used to QEMU monitor (see http://qemu-buch.de/d/Der_QEMU-Monitor). With

the key combination [Ctrl] + [Alt] + [2] leads to the QEMU monitor, with [Ctrl] + [Alt] + [1] back. The creation of virtual machines with different host systems at the URL <http://qemu-buch.de/d/Gast-Systeme> described. Some useful tools for working with QEMU be explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems.

Additional options of QEMU OS / 2 ports

In addition to the numerous QEMU options (see http://qemu-buch.de/d/Anhang/_Startoptionen_von_QEMU_und_KVM) eComstation has the QEMU version for OS / 2 Warp 4 and the *priority option*. For a convenient GUI operation, *idle-priority* option is recommended.

-Priority yet been

No change / use the current priority (default).

-Priority idle

Idle (PRTYC_IDLETIME).

Normal-priority

Normal / Regular (PRTYC_REGULAR)

High-priority

High (PRTYC_FORGROUND SERVER)

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_QEMU_unter_eComstation "

This page has been accessed 6039 times. This page was last updated on 1 November 2010 at 10:27 clock changed. Content is available under GNU Free Documentation License 1.2 .

QEMU for DOS, FreeDOS HX DOS Extender

(Link to this page as [[QEMU-KVM-Buch / QEMU for DOS]])

<<< | # # # | >>> | English

QEMU for DOS

QEMU can run under DOS extender with the HX DOS. The HX DOS Extender (<http://www.japheth.de/HX.html>) allows some Microsoft Windows 32-bit programs written for DOS to run. The HX DOS Extender subject to any specific software license, but is freely available. are a result of the limited possibilities of DOS, there are many restrictions on QEMU. Neither DOS nor the HX DOS Extender support swap files. This virtual memory is at most as large as the physical memory. For example, if a guest system 128 MB of RAM required, it is the host system at least over 170 MB of RAM. The execution speed of QEMU is not under DOS high for several reasons. First, the accelerator KQEMU is not supported, the other is no graphics acceleration. In addition, the DOS is usually run on tired hardware. Using QEMU for DOS can therefore operate only guest systems with very low resource requirements. Since DOS does not support multitasking, QEMU can launch only one instance. Many options of QEMU (http://qemu-buch.de/d/Anhang/_Startoptionen_von_QEMU_und_KVM) are not or only partially available. DOS is certainly not an ideal host system for QEMU. What are we to such a gift with old laptop with just 300 MB of RAM do otherwise? ;-)

For these examples, FreeDOS 1.0 (<http://www.freedos.org/freedos/files/>) installed. The HX DOS Extender also works on other versions of DOS such as DR-DOS 7.03 (<http://www.drdos.net/download.htm>). Descriptions to install FreeDOS and DR-DOS can be found at http://qemu-buch.de/d/Gast-Systeme/_x86-Architektur/_DOS-%2C_Windows_und_Verwandte. The support for long file names (DOSLFN) must be enabled. FreeDOS 1.0 DOSLFN is loaded via *autoexec.bat*.

```
C: \> DOSLFN 1h
```

FreeDOS 1.0 contained in the HX DOS Extender is not current. Therefore *HGXGUI* are the latest version of the ZIP archives from the URL and *HXR* (<http://www.japheth.de/HX.html>) download. Both archives will be in *C:* unpacked *hx*. Furthermore, the *msvcrt.dll* file from <http://www.dll-files.com> download and in *C: hx \ bin \ store*. the *autoexec.bat* file in *C* is the *PATH* variable with the *path: \ hx \ bin* added.

```
set PATH = c: \ hx \ bin,% PATH%
```

QEMU 0.9.1

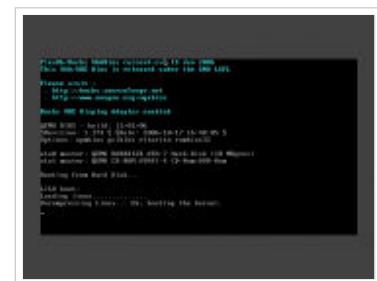
Download: <http://qemu-buch.de/download/qemu-0.9.1-windows.zip>

According to the compatibility list (<http://www.japheth.de/HX/COMPAT.TXT>) QEMU can only up to version 0.9.1 the HX DOS Extender to operate. This is not a great disadvantage, since many options anyway QEMU can be used under DOS. The ZIP archive of the Windows version of QEMU is to download and in *C: \ qemu* to unpack. Then is to change the directory to *qemu*. In order to start QEMU, *qemu* command is always *dpmild32* indicate to the front. The options of QEMU can you get the list with the following command.

```
C: \> dpmild32 qemu - help
```

To scroll the display one page, the following command is used.

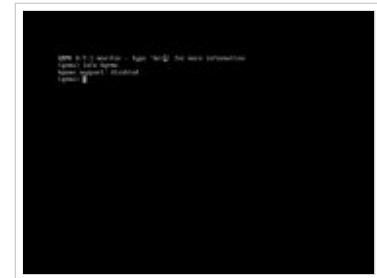
```
C: \> dpmild32 qemu - help | more
```



QEMU for DOS: The virtual machine boots.



QEMU from DOS to Linux as a guest.



QEMU for DOS: The QEMU monitor.

Since QEMU launch a graphical output needed for is dpmild32 with the option to *add-g*. In the ZIP archive of QEMU is the image *linux.img* with minimal Linux system included. The guest system on this image is started with the following QEMU options. *The-L* refers to the directory containing the BIOS file. Since the accelerator KQEMU not available in DOS, use the *no-kqemu* is to *disable*. *With-no-reboot* will prevent the instance to shut down the host system reboots again. As a VGA graphics card is emulated (*-std-vga*). *The-hda* defines the image as a virtual hard disk. With *m-MB*, the memory for the virtual machine defined.

```
C: \> cd qemu
C: \>-g dpmild32 qemu-L. -No-kqemu-no-reboot-std-vga-hda-m 8 linux.img
```

With Linux, the minimum *exit* end.

```
Host ~ # exit
```

This is always the type does not have many options, you can batch-file named *Q.BAT* and create a following content. The value *after-m* is determined experimentally and is about 3 / 4 of the memory of the host system amount. FreeDOS under the program can determine the free space of the *mem* help.

```
set QEMU = C: \ QEMU
dpmild32 QEMU-g%% \ qemu-L%% QEMU-no-kqemu-no-reboot-std-vga-m 8% 1% 2% 3% 4% 5% 6% 7% 8% 9
```

The call is then reduced.

```
C: \> q-hda linux.img
```

Using only one image need not be *specified*, the *hda* case.

```
C: \> q linux.img
```

For QEMU package includes the powerful tool *qemu-img*. It is used to create, convert, and encrypt an image file (virtual hard disks) in various formats (see http://qemu-buch.de/d/Anhang/_qemu-img). *Img* is to *call-without-qemu dpmild32 g*) before the command to specify (. Without options, displays *qemu-img* to a support.

```
C: \> dpmild32 qemu-img
```

This is always the type does not have many options, you can batch-file called *Q-IMG.BAT* content and create a following.

```
set QEMU = C: \ QEMU
QEMU dpmild32%% \ qemu-img% 1% 2% 3% 4% 5% 6% 7% 8% 9
```

With *qemu-img*, you can get information about an image's leave. This is the option *info*.

```
C: \> q-img info linux.img
image: linux.img
file format: raw
virtual size: 10M (10485760 bytes)
disk size: 10M
```

Other

For manipulating the running virtual machine is used to QEMU monitor (see http://qemu-buch.de/d/Der_QEMU-Monitor). With the key combination [Ctrl] + [Alt] + [2] leads to the QEMU monitor, with [Ctrl] + [Alt] + [1] back. The creation of virtual machines at the URL <http://qemu-buch.de/d/Gast-Systeme> described. Some useful tools for working with QEMU be explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_QEMU_unter_DOS "

This page has been accessed 3758 times. This page was last updated on 27 September 2010 at 06:24 clock changed. Content is available under GNU Free Documentation License 1.2 .

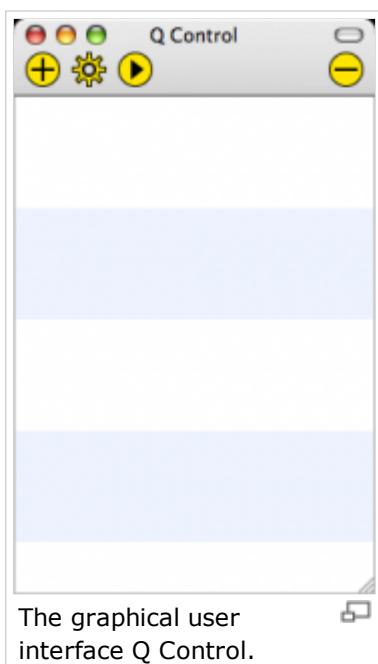
0.13.0 QEMU Mac OS X, Q, QEMU Download build MacPorts macports install, configure make install

(Link to this page as [[QEMU-KVM-Buch / QEMU on Mac OS X]])

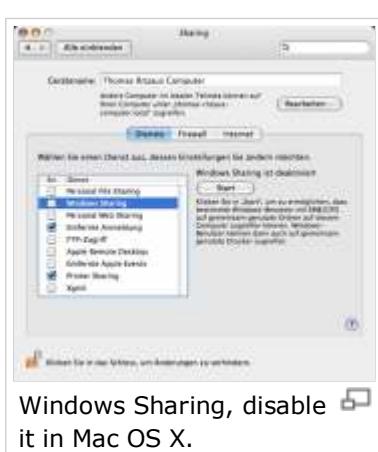
<<< | # # # | >>> | English



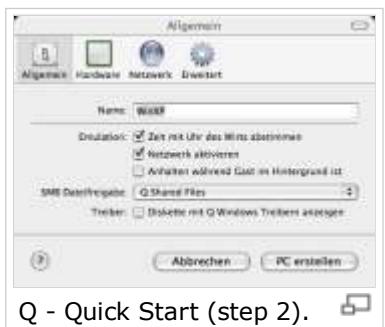
Mac OS X with open disk image.



The graphical user interface Q Control.

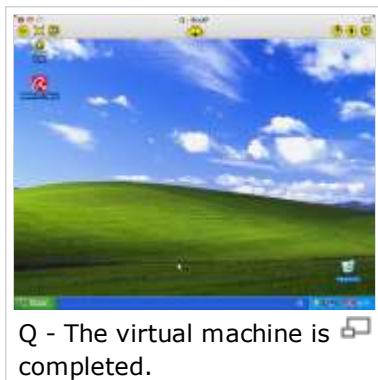


Windows Sharing, disable it in Mac OS X.

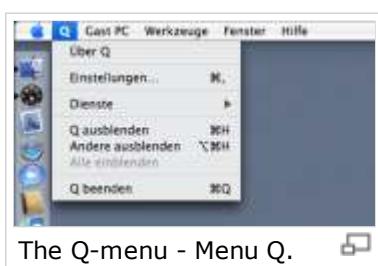




Q - Quick Start (Step 5).



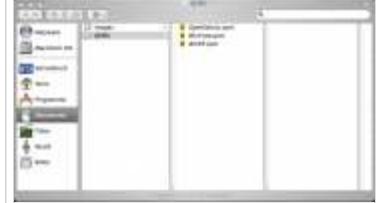
Q - The virtual machine is completed.



The Q-menu - Menu Q.



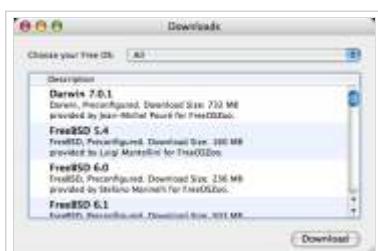
The Q-menu bar - Dialogue with the settings.



The selection of virtual machines via the Finder.



The Q-menu - Menu host PC.



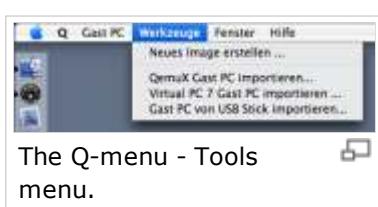
The Q-menu bar - Download complete virtual machines.



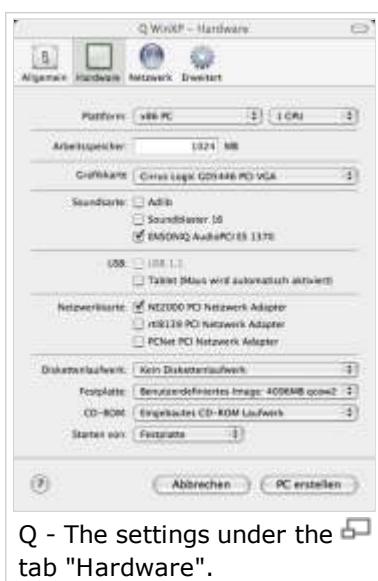
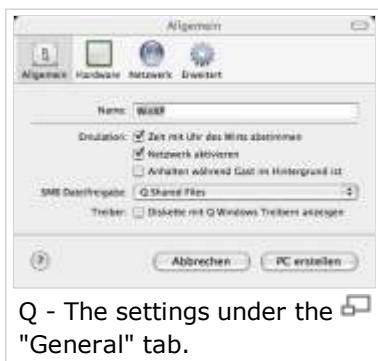
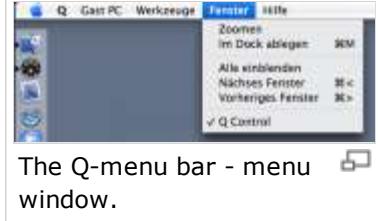
Export a virtual machine on a USB stick.

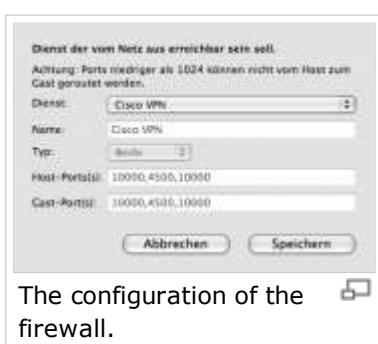
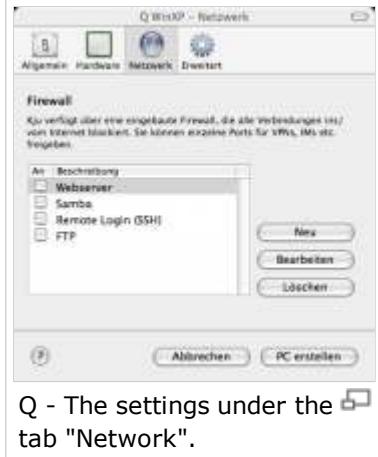


The Q-menu bar - Upgrading a virtual machine on qcows2.



The Q-menu - Tools menu.







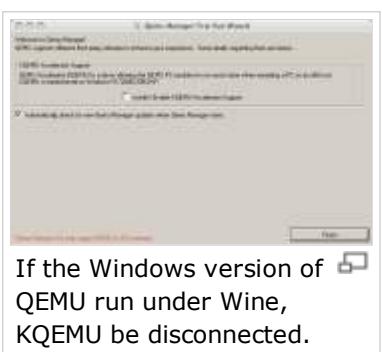
Q Control with a running virtual machine.



Mac OS X - The Fink Commander



Winebuttler - Wine on Mac OS X



If the Windows version of QEMU run under Wine, KOEMU be disconnected.

Contents

1 QEMU on Mac OS X (x86)
01.01 Q - QEMU with a GUI
1.1.1 Installation
1.1.2 Quick Start (Q)
1.1.3 Command Summary
1.1.3.1 The Menu Bar
1.1.3.2 Dialog Q Control
1.1.3.3 Preferences dialog
1.1.3.4 The virtual machine window
2.1 QEMU without GUI
1.2.1 Installation via dmg Image
1.2.2 Installation using Darwin Ports
QEMU 1.2.3 from source compile 0.13.0
1.2.4 Quick Start (command line)
1.3 CDs, DVDs and floppy image as import
4.1 User Space Emulation
1.5 Useful Tools
1.5.1 Fink
1.5.2 WineBottle



The Qemu Manager for Windows on Mac OS X.
Winebuttlr

QEMU on Mac OS X (x86)

Q - QEMU with a GUI

Download: <http://www.kju-app.org>

QEMU on Mac OS X most of the graphical user interface powered by *Q*. *Q* based on QEMU version 0.9.1.

Installation

The following remarks refer to current Apple hardware based on Intel processors. There can be two versions, namely the stable known as *Q-0.9.0a89* and a current development version *Q-0.9.1d118* download. Installation is done with the following steps:

- Download the disk image and appending *Q-*.dmg*.
- Confirm the license conditions.
- Moving the software package *Q* in the program folder *Applications*.

Quick Start (Q)

After the start of *Q* is the control center for virtual machine open. To create a new virtual machine, click the Plus (*New PC*) file icon with the one at the. Then you see a Wizard that asks all the relevant options. In the first dialog of the virtual machine name and the type of the host system is set. In this example, Microsoft Windows XP to be the guest operating system. As a name with one *WinXP* the intended entry field in. As a type of *Windows XP* is selected from the list. After pressing *PC* to create a dialog window with the tabs *General*, *hardware*, networking, and *Advanced*. Under the *General* tab appears in the previous dialogue assigned name of the virtual machine. In *emulation host (the host's time with Clock rate)* of the host system with the time alignment and the network (*network enabled*) enabled.

File sharing between host and guest system is integrated in the *Q* Samba server. In *SMB file sharing* system, the *Q Shared Files* folder on the Exchange host selected. This folder is used by *Q* on the desktop of the logged in user created automatically. This can be accessed in the host system on that folder, a driver in the host system is installed. The folder is drive *Q* in the host system as integrated. The installation file is placed on a drive in the host system. This disk with *Windows drivers* to enable *Q*'s. Once installed, this setting is again to disable. At this point, take a look at the system settings of Mac OS X is required. It is under the *Sharing* tab in the *Windows Services off sharing*. The Mac OS X's built-in Samba server is not compatible with the *Q*'s built-in Samba server.

The *Hardware* tab is used to define the hardware components to be emulated. *Below deck* is to be selected *x86PC*. The number of emulated CPUs is altered. The example should suffice a CPU. Memory should the virtual machine with Microsoft Windows XP attributed abundant, less than 512 MB are not recommended. In *graphics card* is a Cirrus Logic GD5446 PCI VGA selected. The available sound and network cards are accepted. In *USB Tablet* is activated. On a floppy disk drive is dispensable (*no floppy drive*). The hard drive is a new, four gigabytes of image must be created. The installation of the host system from a real CD is due to a bug of Q not possible. Therefore, an ISO image of Windows XP Boot CD can be used. To install the guest system is *starting CD-ROM* set on. After the installation is switched back to *hard drive* on. Under the *Network* tab settings are set to leave. Under the *Advanced* tab there is no need to change. With *PC create* the virtual machine is completed. It appears in the list of available virtual machine and is started by double clicking on the name. The new virtual machine boots from the image of the installation CD and performs the usual operating system installation.

Cheat sheet

The user interface consists of a menu bar, and some dialogs.

The menu bar

The menu *Q* corresponds to a typical Mac OS X application. Here) is information about the program (over *Q*, the basic settings (*settings* ...), the call of some Mac OS *services* (Services), functions to control windows on the desktop (*Q Hide, Hide Others, Show All*) and exit the program (*Exit Q*) is housed. Of particular interest are the *settings*. *Sub-display* to permit the type of graphics output set of virtual machines. Choose from *OpenGL*, *Quartz* and *QuickDraw*. *OpenGL* is a platform-independent programming interface for 3D graphics in the graphics card driver is able to implement. As a result, large parts of the calculation of the graphics output are directly done by the processor on the graphics card. *OpenGL* assumes some of the graphics card emulation of the host PC directly to the graphics processor of the host system. This requires hardware *OpenGL* support each of these. All modern graphics cards that is the case. The CPU of the host system is from the tasks of the graphics output of the host system relieves. *Quartz* is the graphics layer of Mac OS X. It is abstracted from the possibility of actually built-in graphics hardware and provides a uniform application programming interface. Capabilities of the hardware, such as *OpenGL*, transparently embedded there. This type of graphical output is all computers with Mac OS X from. *QuickDraw* as graphics layer of Mac OS 9 is not recommended.

For existing Internet access *Q* will automatically at program start site (*When the program starts checking for updates*) to versions on the new project. *Log in to write console* is activated, messages are useful debug output to the system console. The *console* program (found under the utilities) messages makes them visible. The virtual machines in *Q* are mapped to multiple files. In addition to the disk images of the host PC to the preference files are the options for QEMU. These files are stored by default in a directory structure below the home directory. The default path is / *Users* / <*user*> / *Documents* / *QEMU* is the Finder of documents accessible. For a single user this may well be sufficient. Every other user has access only to its own virtual machine. In order to avoid this, files can be the location of the file system *folder* with *Q* score freely defined. Also, the virtual machines are stored on a central server on the network.

Menu *Guest PC* can create new virtual machines (*New Guest PC*) and edit existing virtual machines (*guest editing PC*) or delete (*Delete host PC*). These functions are also directly accessible via *Q Control*, where they are explained in more detail. Download the complete virtual machines from the internet supported (*PC host of new free.oszoo.org download*). Although only affects operating systems, their licensing conditions, this type of use and disclosure of the permit, the selection is still considerable (see http://qemu-buch.de/d/Gast-Systeme/_FreeOsZoo).

It's *Q*, as the software tool package USB-Stick to export (*to host PC USB Stick export*) to a possible virtual machine, with one and QEMU. The scale package can run on any Intel-based Mac. The installation of QEMU is not needed. Only an emulated in the virtual machine CD-ROM drive is causing problems. the drive in the virtual machine is included, this does not start. Therefore, you should disable before exporting the CD-ROM drive in the hardware settings of the virtual machine.

Since version 0.9.0 the format *qcow2* dominated QEMU virtual hard disks. With *guest PC and compress* will update previous versions created virtual machines on the current state to be related to. From the *Tools* menu, virtual hard disks created. In the dialog under the menu item *New Image ...* create the file size and type of image set. As a type *qcow* are *raw* or optional (see http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Speichermedien/_Image-Formate). Furthermore, virtual machines from Virtual PC 7

and QemuX be imported. QemuX project is the predecessor of Q and is no longer being developed. The import of an exported virtual machine on a USB stick is possible here.

The *Window* menu contains no specific functions of *Q*, there are just some window management features of Mac OS X included. In the *Help* menu, *Q* is the integrated help of reach. This is the *Help Viewer* in Mac OS X appears in the. The help is in English only and not on the actual state of the program.

Dialogue Q Control

At program startup dialog opens *Q Control*. About the icons can be controlled by the virtual machines.

-  Wizard to generate a new virtual machine.
-  Settings of a virtual machine.
-  Start a virtual machine.
-  Deleting a virtual machine.
-  Import virtual machines from Virtual PC.
-  Import virtual machines from QemuX.

The appearance of the icons (image only, image and text, text only) can be changed. This is done with [Ctrl] + click or right-click on the icon bar, the desired change. In the list all available virtual machines. Started virtual machines have a small preview. By clicking on the plus icon to a wizard called *Q*, which creates a new virtual machine. The first dialog prompts you to name the new virtual machine and selecting the desired guest operating system. Guest operating systems here are DOS, various Windows versions and a standard *Q* unspecified guest selected. After the activation of *PC to create* dialog is automatically displayed for the other settings *General*.

Settings dialog

Here, the name and basic settings of the virtual machine can be changed.

Name	Name of the virtual machine.
Time with the clock rate of the host	Especially with Microsoft Windows as guest system taking over the system time from the host system is recommended. Unix-like guests get their system better time with an NTP server.
Network Enable	Releases the network functionality. To make the access from the host system to the Internet. <i>Q</i> also allows it to share files between host and guest. If no Internet access on the host system exist, this option will automatically disappear.
Stopping in the background while guests	To save resources on the host system, the emulation of the host system is stopped when other applications are running on the host system in the foreground. On modern multi-processor systems, this option is off, as enough computing power is available.
SMB file sharing	<i>Q</i> offers the possibility of file sharing between host and guest on an integrated Samba Server (SMB). <i>Q</i> This sets the default user's desktop to a folder on <i>Q Shared Files</i> . Used the built in QEMU Samba server is only when the built-in Mac OS X Samba server is disabled. In the system settings to the service to disable Windows Sharing under <i>Sharing</i> .
Disk with Q's Windows drivers	This option appears in a Windows guest up a drive that contains a driver for data exchange between host and guest.

Table: Q - The settings under the "General" tab.

Platform	Below deck, the processor is chosen to be emulated. At present, the x86 PC is stable. The x86-64 PC is experimental. Also adjusted the number of emulated CPUs.
----------	---

Memory	The memory for the host system is configured here.
Video Card	This option can be set to the emulated video card. For host systems with graphical user interface, the <i>Cyrus Logic</i> card to be selected. For pure console systems sufficient <i>standard VGA</i> .
Sound Card	Here, three different models of sound maps.
USB	The support of USB devices, the host is not possible. Only the support for the virtual tablet is activated.
NIC	There are three emulated network cards to choose from.
Floppy / hard disk / CD-ROM	This mass storage can be defined. In addition to the existing host devices (floppy / CD / DVD-ROM) image files are selected. Hard drives get mounted as image files.
Starting	The BIOS of the emulated PC supports booting from the hard disk, floppy disk or CD / DVD-ROM. In the version 0.9.0.d82 does booting from the CD / DVD-ROM is not the host. It is therefore the disc to import as an image.

Table: Q - The settings in the Hardware tab.

The network settings are based on QEMU's built-in firewall. About *New* protocols are enabled. This service is to add a label to indicate the type of communications (TCP, UDP or both) and the ports for host and guest (Port-redirect).

Hard drive 2 .. 4	With these options up to three disk images into the virtual machine can be integrated.
Windows 2000 Installation Help	The installation of Microsoft Windows 2000 fails because of an allegedly full hard disk. This can be achieved with the <i>hack-around win2h</i> . After installation, this option is to disable it.
Settings for Linux guest systems	These options Linux kernel will be launched directly (see http://qemu-buch.de/d/Spezielle_QEMU-Optionen).
QEMU options	Additional options for QEMU that in Q are not adjusted to be involved.

Table: Q - The settings under the Advanced tab.

The virtual machine window

The started virtual machines are each displayed in a window. It is possible to switch to full screen mode. This is to click on the appropriate icon in the toolbar or the [Ctrl] + [Alt] to activate + [F]. In full screen mode you can switch to the shortcut keys [Cmd]+[<] or [Cmd]+[>] between the virtual machine and the Mac OS X desktop.

	This allows both the CD drive of the host, as well as various CD images show in the current guest.
	Switches from window to full screen mode.
	Creates a screenshot of the virtual machine and present it in PNG format from the desktop.
	Here are images of floppy disks can be in the running guest system show. There are two drives (floppy A, disk B) are involved.
	Stops execution at the virtual machine. The next push execution continues.
	Performs a reset (cold start), the virtual machine.

Sends the key combination [Ctrl] + [Alt] + [Del] to the virtual machine.



Off the virtual machine, regardless of declines.



Show the CPU utilization and I / O activity (disk, CD / DVD-ROM).

QEMU without GUI

These versions are configured via the command line and Q can be installed in addition. The command line is provided through the *Terminal* (*Macintosh HD, Applications, Utilities*).

Installation by dmg Image

Download: <http://www.puredarwin.org/downloads>

The installation image *qemu-*.dmg* is download and under */ opt / local / bin / qemu* to install.

installation via Darwin ports

Download: <http://qemu.darwinports.com>

MacOS X is based on Darwin. Darwin is a Unix BSD roots. Similar to the ports of FreeBSD (see http://qemu-buch.de/d/QEMU_unter_BSD) is software for MacOS X, Darwin ports are also installed. Necessary to the installation of the Darwin Ports. The installation package can do so by the URL <http://darwinports.com> download and install. Subsequently, the Darwin Ports to be updated.

```
Host ~ $ sudo port-d selfupdate
```

After ports are examined with the command *port search*. The following command lists all ports on *qemu* to.

```
Host ~ $ port search qemu
```

The port is the *qemu* with the following command to */ opt / local / bin / qemu*.

```
Host ~ $ sudo port install qemu
Password :*****
```

QEMU 0.13.0 compile the sources from

Downloads: <http://download.savannah.gnu.org/releases-noredirect/qemu/>

Compiling from source allows you to install the latest version of QEMU. In the Mac user tinkers with a Unix BSD roots. Apple has indeed rebuilt in some places strong, but the adaptation of many open source projects on Mac OS X is possible. As compiler uses Apple for example, the gcc suite. The *Xcode* development environment, the resources required to produce their own programs contains all heard with Mac OS X. However, it is not installed by default. To install the package *XcodeTools.mpkg* that on the Mac OS X *Install Disc 1* in the *Xcode Tools* folder is located.

To compile the *terminal* is to start (*Macintosh HD, Applications, Utilities*). Compile it with normal user rights. Initial installation requires root privileges. First, the source packages for QEMU be downloaded and unpacked. *Make install* directories are the files created in the appropriate copies.

```
Host ~ $ mkdir-p ~ / source
Host $ cd ~ / source
Host ~ $ wget \
http://download.savannah.gnu.org/releases-noredirect/qemu/qemu-0.13.0.tar.gz
```

```
Host ~ $ tar xzvf qemu-0.13.0.tar.gz
Host ~ $ qemu-0.13.0
Host ~ $ . / Configure-disable-darwin-user - disable-bsd-user - enable-cocoa
Host ~ $ make
Host ~ $ sudo make install
```

A look at the Date *qemu-doc.html* is recommended. The compiled version is under */usr / local / bin /*. To test QEMU is called.

```
Host ~ $ / usr / local / bin / qemu-version
QEMU PC emulator version 0.13.0, Copyright (c) 2003-2008 Fabrice Bellard
```

Quick start (command line)

The QEMU boot options are on all operating systems equal. To demonstrate the key options in the example below, a virtual machine using shell commands will be generated. The call to QEMU on the command line has the advantage that error messages are displayed immediately. When creating a virtual machine the following steps:

1. First, a virtual disk (image file) is created.
2. After that QEMU is started with this virtual hard disk and the installation media. The virtual machine is configured with boot options.
3. The operating system, as installed in a real machine. On the support of the emulated hardware is respected.

For this example (which was the operating system ReactOS <http://www.reactos.org/de/>) selected as the guest. There is created a virtual hard disk. This is done with the package contained in the QEMU *qemu-img* tool and *create* the parameters. Given the name and size of the virtual disk to be indicated. With *the-f* format, the preset.

```
Host ~ $ qemu-img create-f qcow2 ReactOS.img 1G
```

From the ReactOS website is the image of the installation CD to download and unpack. Is called QEMU *qemu* with the command. The configuration of the virtual machine is in options on the command line. To install the virtual machine is started from the installation CD. Whose name is entered after the *option-cdrom*. The name of the created virtual hard drive on the *option-typed hda*. This virtual computer from the CD will boot the virtual, *d* is the *option-boot* responsible.

```
Host ~ $ qemu-hda ReactOS.img-cdrom-boot d ReactOS.iso
```

It will start the virtual PC in a window. Installing ReactOS is started by clicking in the window and pressing any key. First, the German keyboard is selected. Thereafter, the virtual disk is partitioned and formatted. The target directory for the installation of the ReactOS files need not be changed. After the boot loader is installed, the default option *Install bootloader on the harddisk (MBR) holds*. Then the virtual machine must be restarted and the installation continues graphically. In order to boot from the hard disk, *c* is the *option-boot* to apply.

```
Host ~ $ qemu-hda ReactOS.img-cdrom-boot c ReactOS.iso
```

To boot without the CD image but with the right system time, the command applies. This QEMU configured via the integrated DHCP server, the network settings of the host system.

```
Host ~ $ qemu-localtime ReactOS.img
```

See also <http://qemu-buch.de/d/Quickstart>.

DVDs and floppy disks as image import CDs

A CD is loaded by the system to */Volumes / <name>* included automatically. The name of the CD will be indicated in the Finder. In this example, the *W2P_DE*. To read the CD first the appropriate device is to be identified. In Mac OS X Device files are displayed *dev /* under. Interestingly, the devices */dev / disk **, since these disks and CD drives used to be. No CD is inserted, for example, the following entries on the

subject.

```
Host ~ $ ls / dev / disk *
/ Dev/disk0 / dev/disk0s1 / dev/disk0s2
```

This is the first hard drive *disk0*. This here contains two partitions, *disk0s1* and *disk0s2*. If a CD is to be */dev* automatically inserted a new device.

```
Host ~ $ ls / dev / disk *
/ Dev/disk0 / dev/disk0s1 / dev/disk0s2 / dev/disk1 / dev/disk1s0
```

The CD drive is here */dev/disk1s0* raised with. Involved in the state program the system refuses the *dd* to access the drive. So first, the drive is again suspended. The necessary administrative rights gives it with *sudo*.

```
Host ~ $ sudo umount / Volumes/W2P_DE
```

Thereafter, *dd* the contents of the CD to read and write in the ISO image.

```
Host ~ $ dd if = / dev/disk1s0 of = / pdb / data / cd-image.iso
```

To conclude, one should embed the CD again, otherwise the ejection of the CD is no longer located on the desktop icon or use the Finder works.

```
Host ~ $ sudo mount_cd9660 / dev/disk1s0 / Volumes/W2P_DE
```

User Space Emulation

In Mac OS X QEMU is also the user-space emulation supported (see # http://qemu-buch.de/d/Spezielle_QEMU-Optionen userspace emulation).

Useful Tools

Fink

Fink is a port of the comfortable package tools *dpkg* and *apt-get* from the Debian world. In addition to the command line tools with the *Fink Commander* is a graphical user interface included in the package too. Fink is based on the URL <http://sourceforge.net/projects/fink/files/>. To install the following steps are necessary.

- Download the disk image and appending *Fink-*- Intel Installer.dmg*.
- Confirm the license conditions.
- Select the hard disk.

The installation package is run. The programs end up where in the applied tree */sw*. The shell variable *\$PATH* is set automatically to */sw/bin* and */sw/sbin* added. Thus the installed via Fink programs are found automatically by the system. Installing packages is as for Debain / Ubuntu.

```
Host ~ $ sudo apt-get install wget unzip
```

WineBottle

WineBottle is the port of Wine (http://qemu-buch.de/d/Anhang/_Nützliche_Tools/_Wine) for Mac OS X. Wine (Wine Is Not an Emulator) is a Windows-compatible runtime environment for POSIX-compatible operating systems. This makes it possible for Microsoft Windows operating systems, programs compiled under BSD Unix (* BSD, Solaris, Mac OS X) to let run, and Linux. Wine is not an emulator or a virtualization solution. Wine with the system to the appropriate Windows APIs will be extended. Wine is open source (GNU LGPL) and does not require a Windows license. A WineBottle documentation can be found at the URL <http://wiki.winehq.org/WineBottler>. WineBottle used to be called Darwin. Darwin under the PowerPC architecture uses the userspace emulation QEMU. To install on Mac OS X WineBottle), the image (. Dmg downloaded and integrated the software package in the Applications folder (*Applications*) moved. More information can be obtained at the URL <http://qemu-buch.de/d/Anhang>

/_Nützliche_Tools/_Wine .

[<<< | # # # | >>>](#)

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_QEMU_unter_Mac_OS_X "

This page has been accessed 8985 times. This page was last updated on 17 October 2010 at 17:11 clock changed. Content is available under GNU Free Documentation License 1.2 .

QEMU in BSD (Solaris, OpenSolaris, Illumos, FreeBSD, OpenBSD, NetBSD), Win4BSD, KQEMU

(Link to this page as [[QEMU-KVM-Buch / BSD under QEMU]])

<<< | # # # | >>> | English

Contents	
QEMU under a BSD (Solaris, OpenSolaris, Illumos, FreeBSD, OpenBSD, NetBSD)	
1.1 Installation	
1.1.1 Solaris, OpenSolaris and Illumos	
1.1.1.1 x86/64-Architektur	
1.1.1.2 SPARC	
FreeBSD 1.1.2	
PC-BSD 1.1.3	
NetBSD 1.1.4	
2.1 Quick Start	
1.3 User Space Emulation	
4.1 Win4BSD and Win4Solaris	



OS / 2 and Microsoft Windows 98 to run in QEMU on Solaris 10 on SPARC processor architecture.

QEMU on BSD (Solaris, OpenSolaris, Illumos, FreeBSD, OpenBSD, NetBSD)

Installation

Solaris, OpenSolaris and Illumos

x86/64-Architektur

For the x86/64-Architektur package is a QEMU on the website <http://www.thoughtwave.com/downloads.html> the point *QEMU v0.9.1 with MTools 3.9.10 for Solaris 10* for download at. After the download unpack the file.

```
Host ~ # bzip2-d-0.9.1-THOTqemu_20,080,113-universal-solaris10.bz2
```

Subsequently the tool of this package installed *pkdadd*.

```
Host ~ # pkgadd-d THOTqemu-0.9.1-20080113-universal-solaris10
```

It is to put the following link so that the *library* is found *1.2.so.0-libSDL*.

```
Host ~ # ln-s /opt/thoughtwave/lib/libSDL-1.2.so.0 \
           /Lib/libSDL-1.2.so.0
```

After the installation is called QEMU, indicating the path.

```
Host ~ $ /opt/thoughtwave/bin/qemu
```

So you do not specify the path must be accessible, the search path is extended as follows.

```
Host ~ $ export PATH=$PATH:/opt/thoughtwave/bin/
```

The optional accelerator *KQEMU 0.11.1* can be used to QEMU. KQEMU is to compile from source. This requires only the tools are installed. A simple installation is possible with the tool *pkgutil*. This tool to load from the URL <http://www.blastwave.org> download and install it with *pkgadd*.

```
Host ~ # pkgadd-d pkgutil_sparc.pkg
Host ~ # cp-p /opt/csw/etc/pkgutil.conf.CSW /etc/opt/csw/pkgutil.conf
```

So you do not specify the path to *pkgutil* must, you can expand the search path.

```
Host ~ # export PATH=$PATH:/opt/csw/bin/
```

First, the package list is updated.

```
Host ~ # pkgutil -catalog
Host ~ # pkgutil -install gnupg textutils
```

It will install the necessary tools.

```
Host ~ # pkgutil -gmake install gcc4 gnustep_make wget
```

Now site is the tar archive with the accelerator of the *KQEMU* <http://opensolaris.org/os/project/qemu/downloads/> downloaded and unpacked.

```
Host ~ # bzip2-d \kqemu-1.3.0pre11_sol10FCplus_20070214_src_and_bins.tar.bz2 host ~ # tar xvf \kqemu-1.3.0pre11_sol10FCplus_20070214_src_and_bins.tar
```

In the created directory compile the sources.

```
Host ~ # cd kqemu-1.3.0pre11__osol20070214
Host ~ # ./Configure
Host ~ # qmake
```

The generated module *kqemu-solaris-i386* is to rename the kernel.

```
Host ~ # mv kqemu-solaris-i386 kqemu
```

Comprising the steps of the kernel module is installed.

```
Host ~ # /usr/sbin/install-f /usr/kernel/\drv
      755-m-u root-g sys kqemu
new owner is root
kqemu installed as /usr/kernel/drve/kqemu
Host ~ # /usr/sbin/install-f /usr/kernel/\drv
      644-m-u root-g sys kqemu.conf
new owner is root
kqemu.conf installed as /usr/kernel/drve/kqemu.conf
```

Module is activated with the *add_drv*.

```
Host ~ # add_drv kqemu
```

As a control, the *modinfo* command applied.

```
Host ~ # modinfo | grep kqemu
215 f9c67000 1a608 207 1 kqemu (kqemu accelerator v0.2)
```



On Solaris 10 running Windows NT in QEMU.



AQEMU - A GUI for QEMU.



Win4BSD - Microsoft Windows XP as a guest.

After the download unpack the file.

```
Host ~ # bzip2 -d 0.9.1-THOTqemu_20,080,113-universal-solaris10.bz2
```

Subsequently the tool of this package installed *pkdadd*.

```
Host ~ # pkgadd -d THOTqemu-0.9.1-20080113-universal-solaris10
```

It is to put the following link so that the *library* is found *1.2.so.0-libSDL*.

```
Host ~ # ln -s /opt/thoughtwave/lib/libSDL-1.2.so.0 \
           /Lib/libSDL-1.2.so.0
```

After the installation is called QEMU, indicating the path.

```
Host ~ $ /opt/thoughtwave/bin/qemu
```

So you do not specify the path must be accessible, the search path is extended as follows.

```
Host ~ $ export PATH=$PATH:/opt/thoughtwave/bin/
```

FreeBSD

Software is available for FreeBSD ports and packages. Ports include the source code of the software. They are compiled during installation. Ports are often more current than packages. See the FreeBSD Handbook at the URL http://www.freebsd.org/doc/de_DE.ISO8859-1/books/handbook/. QEMU to relevant ports are located in the *emulators* in the port tree. In the subdirectory *qemu* and the installation directory of QEMU *kqemu-kmod* KQEMU started the installation of the. From QEMU 0.12.0 KQEMU is no longer supported.

```
Host ~ # cd /usr/ports/emulators/qemu
Host ~ # make install clean
Host ~ # cd /usr/ports/emulators/kqemu-kmod
Host ~ # make install clean
```

It is also possible to install QEMU as a package. Package are ready-compiled software packages. Click to install QEMU and KQEMU as packages:

```
Host ~ # pkg_add -r qemu
Host ~ # pkg_add -r kqemu-kmod
```

KQEMU is loaded as a kernel module.

```
Host ~ # kldload kqemu
Host ~ # kldload aio
```

With the command *kldstat* is checked whether the modules were loaded, too.

```
Host ~ # kldstat | grep kqemu
```

Thus after each restart the modules and loaded *kqemu aio*, *rc.conf* is this line in the startup script / *etc* / write.

```
kqemu_enable = "YES"
```

As GUI AQEMU is recommended (see http://qemu-buch.de/d/Managementtools/_AQEMU).

```
Host ~ # cd /usr/ports/emulators/aqemu
Host ~ # make install clean
```

On FreeBSD it is also possible AQEMU to install as a package.

```
Host ~ # pkg_add -r aqemu
```

On the console command is started the GUI with the *aqemu*.

```
Host ~ $ aqemu
```

PC-BSD

Download: <http://pbidir.com/bt/pbi/77/qemu>

Installing QEMU under FreeBSD derivative PC-BSD is possible with the PBI System (Push Button Installer) with a few mouse clicks. It is downloaded, the PBI package for QEMU and installed with the PBI launcher. The PBI package also includes QEMU AQEMU. The files are */usr/Programs/QEMUX.xx_x* / stored in. If the message *.PBItmp.pbistart* / not found, you have a PBI package for the wrong PC-BSD version downloaded.

NetBSD

NetBSD source software in package (*pkgsrc*) available for. If not already have the package source system to be installed. This simplifies the download of the packages of the current industry as a tar archive is described. The package system can use the *Software Update Protocol* (SUP) or *Concurrent Versions System* (CVS), downloaded and be kept up to date.

```
Host ~ # cd /usr
Host ~ # ftp ftp://ftp.netbsd.org/pub/pkgsrc/current/pkgsrc.tar.gz
Host ~ # tar xzvf pkgsrc.tar.gz
```

QEMU is installed.

```
Host ~ # cd /usr/pkgsrc/emulators/qemu
Host ~ # make install clean
```

Quick Start

Quick Start section in the instructions is the creation of a virtual machine using shell described (see <http://qemu-buch.de/d/Quickstart>). In the section AQEMU AQEMU the GUI described (see http://qemu-buch.de/d/Managementtools/_AQEMU).

User Space Emulation

BSD supports QEMU, version 0.10.0 the userspace emulation (see [# userspace emulation](http://qemu-buch.de/d/Spezielle_QEMU-Optionen)).

Win4BSD and Win4Solaris

Win4BSD and Win4Solaris are commercial products of Virtual Bridges, Inc., based on QEMU. This means that the Microsoft Windows versions 2000 and XP or run individual Windows applications. The non-commercial use of Win4BSD is free. Since this version is only a 32-bit application is for 64-bit hosts the original version of QEMU (64-bit) recommended. A description can be found at the URL http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulatoren/_Win4 .

```
<<< | # # | >>>
```

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_QEMU_unter_BSD"

This page has been accessed 7755 times. This page was last updated on 27 September 2010 at 18:37 clock changed. Content is available under GNU Free Documentation License 1.2 .

Guest systems under QEMU and the Kernel-based Virtual Machine

(Link to this page as [[QEMU-KVM-Buch / guest systems]])

<<< | # # # | >>> | English

Guest Systems

Official QEMU OS Support List: <http://www.claunia.com/qemu/>

KVM Guest Support Status: http://www.linux-kvm.org/page/Guest_Support_Status

In this chapter some sample installations of guest systems are described. Other examples can be found at the URL <http://qemu-buch.de/d/Gast-Systeme>. The information for the examples Information on memory and disk size are recommendations and can be adjusted according to personal preference. For older operating systems often have the RAM and hard drive size will be limited. Also, additional options have to be that emulate the old hardware. Newer operating systems require more resources. Note the relevant provisions of the license to install a guest system.

FreeOsZoo
x86

- DOS, Windows and Related
- BSD Unix systems
- Linux
- Unix-like operating systems
- OpenVMS-like operating systems
- Exotics
- Hypervisor

SPARC

- ARM processor architecture
- MIPS processor architecture
- Coldfire processor architecture
- PowerPC architecture
- SuperH architecture

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme "

This page has been accessed 6511 times. This page was last updated on 3 March 2010 at 17:15 clock changed. Content is available under GNU Free Documentation License 1.2 .

FreeOsZoo download the virtual machine, Free OS Zoo Live Virtual Machines in Web Browser

(Link to this page as [[QEMU-KVM-Buch / guest systems / FreeOsZoo]])

<<< | # # # | >>> | English

FreeOsZoo - Download Virtual Machines

Website: <http://www.oszoo.org>

The FreeOsZoo will not be updated.

installed and configured virtual machines for QEMU and KVM, the project FreeOsZoo available. Although this only affects operating systems, their licensing conditions, this type of use and disclosure of stipulate the selection at the URL <http://www.oszoo.org/ftp/images/> is still considerable. The download of virtual machines is possible with BitTorrent. We choose to Image from one and loads the appropriate file (. Torrent) down. For most desktop systems, while the BitTorrent client starts automatically. The BitTorrent protocol is blocked by some firewalls. The archives with the images, therefore, as a zip or tar file again. Windows users can unpack with 7-zip archives. In the archive are located next to the image of a README file, the information about the options for using the system and provides passwords to the host system. Anyone can add their own virtual machines and help the FreeOsZoo that the selection is expanded. First, create a large image in a compressed format qcow and install the guest system. Alternatively, to convert an image with an installed operating system in a compressed format qcow. As an example, here is a ReactOS image. The filename of the image should identify the installed operating system and version.

```
Host ~ $ qemu-img convert-c-O qcow \
      ReactOS.img ReactOS 0.3.11.img
```

It is to create a directory, see its name, the operating system and version does.

```
Host $ mkdir ~ ReactOS 0.3.11
```

The image file is to move to the directory.

```
Host ~ $ mv 0.3.11.img ReactOS ReactOS 0.3.11
```

Furthermore, in the directory is a *README.txt* file to write to, the information includes the following: *Name, Author, QEMU command, login, and Description*. The name is the name of the image file after login are given when required user name and password.

```
Name: ReactOS 0.3.11.img
Author: Robert Warnke
QEMU command: qemu-hda ReactOS 0.3.11.img
Login: no password
Description: ... www.reactos.org
```

Thereafter, a zip or a tar archive is to create this directory.

```
Host ~ $ tar cvf 0.3.11.tar ReactOS ReactOS 0.3.11
```

To this archive is a torrent file with the URL <http://www.oszoo.org:6969/announce> as trackers produce.



Free Live OS Zoo - operating systems without installing the Web-browser testing.



The Free Live OS zoo with ReactOS.

Here is an example of the BitTorrent client on Ubuntu Linux (packages *bittorrent* and *bittornado*).

```
Host ~ $ btmakemetafile http://www.oszoo.org:6969/announce \
ReactOS 0.3.11.tar
```

The file is *0.3.11.tar.torrent* ReactOS (Marinelli AT cs.unibo.it) to send to Stefano. In response you get the confirmation that the torrent file has been released on the tracker. The upload is done with a BitTorrent client. First, this is the torrent file and then select the tar or zip archive. To make known to the newcomer, with a note <http://www.oszoo.org/blog/> write.

Free Live OS Zoo - Virtual Machines in Web Browser

Website: <http://connessi.v2.cs.unibo.it:8880>

In the experimental stage is the project Free Live OS Zoo (http://www.oszoo.org/wiki/index.php/Free_Live_OS_Zoo) that testing of operating systems in the Web browser supports it. This is run on the servers of this project, request a virtual machine under QEMU. A VNC client as a Java applet that allows communication with the machine. In the URL you choose from a virtual machine and the keyboard layout. Please click on the button *Load image* A QEMU starts on the server. About the web browser log into into the system.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_FreeOsZoo "

This page has been accessed 5954 times. This page was last updated on 27 September 2010 at 18:38 clock changed. Content is available under GNU Free Documentation License 1.2 .

Guest systems x86

(Link to this page as [[QEMU-KVM-Buch / guest systems / x86]])

<<< | # # # | >>> | English

x86

Some host systems to connect with the installation if KQEMU or KVM is active. These problems are avoided by installing the *option-no-kqemu* (QEMU older versions) or *-no-kvm* uses. After installation, KQEMU (*-kernel-kqemu*) or usually be re-enabled KVM. Not all x86-guest systems that can be run under QEMU boot, even under KVM.

Many x86 operating systems can be installed www.netboot.me with the service. This is the CD image download and integrate as a boot device. In the boot menu, you simply select the desired operating system.

```
Host ~ $ wget http://static.netboot.me/gpxe/netbootme.iso  
Host ~ $ qemu-hda-cdrom Platte.img netbootme.iso
```

- DOS, Windows and Related
- BSD Unix systems
- Linux
- Unix-like operating systems
- OpenVMS-like operating systems
- Exotics
- Hypervisor

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_x86-Architektur "

This page has been accessed 4420 times. This page was last updated on 27 September 2010 at 18:39
clock changed. Content is available under GNU Free Documentation License 1.2 .

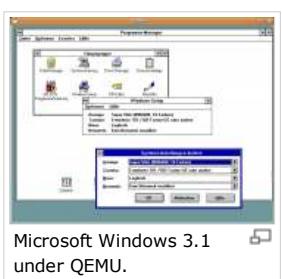
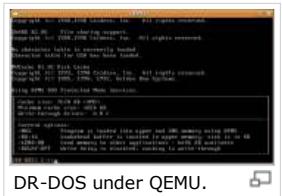


The CD image from netboot.me.

Guest systems: x86, DOS, Windows and relatives, Microsoft Windows 7 Ultimate, download iso installation 0x800703e6

(Link to this page as [[QEMU-KVM-Buch / guest systems / x86 / DOS, Windows and relatives]]))

<<< | # # # | >>> | English





Microsoft Windows 3.1
with Calmira as a GUI.



Microsoft Windows 95 on
QEMU.



Microsoft Windows 98 on
QEMU.



In this case, Microsoft
Windows NT QEMU runs on a
QEMU instance, was started
in the mini-Linux.



Microsoft Windows 2000
under QEMU.

Contents	
1 DOS, Microsoft Windows and related	
1.1 CP/M-86	
2.1 DR-DOS 7.03	
FreeDOS 1.3 1.0	
4.1 DESKWORK	
1.5 OS / 2 Warp 4	
1.6 Microsoft Windows (for Workgroups) 3.xx	
1.7 Microsoft Windows 95	
1.8 Microsoft Windows 98	
Microsoft Windows NT 1.9	
1:10 Microsoft Windows 2000	
1:11 Microsoft Windows XP	
1:12 Microsoft Windows Home Server	
1:13 Microsoft Windows Vista	
1:14 Microsoft Windows 7 Enterprise (32 -, 64-bit)	
1:15 ReactOS 0.3.12	

DOS, Microsoft Windows and related

MS-DOS (Microsoft Disk Operating System) took over many of the concepts of CP / M and some of the concepts of Unix. It was in the late 1980s and early 1990s, the dominant operating system for a single computer. The now obsolete and withdrawn from the market Windows versions 1.0-3.11, 95, 98 and Me were dependent on DOS. Microsoft Windows NT and Microsoft operating systems based on them not to build on MS-DOS and DOS software can only run limited. It is found only a 16-bit compatibility layer. Direct access to real hardware is prevented for reasons of system stability. Microsoft DOS and Windows versions are commercial products. A free download is not possible. Note the respective license.

QEMU emulates the video card Cirrus Logic GD5446. All Windows versions from Windows 95 to recognize this graphics card. For higher resolutions (> 1280x1024x16) is Microsoft Windows XP, the option to use *std-vga*. For optimum performance, a 16-bit color depth can be configured in the host and guest. In a slow host system using Microsoft Windows 95 or Microsoft Windows 2000 is recommended as a guest. For data exchange with a Unix-/Linux-Host Samba is used. With newer QEMU / KVM versions will change the emulated hardware. QEMU is replaced or the Kernel-based Virtual Machine upgrade, some Microsoft Windows systems must be reset. Thus, the emulated hardware does not change the machine type to the QEMU version is bind. 0.11.0 For example, the machine type of the QEMU version *pc-0.11-1-M* set to.

DOS versions use the CPU HLT commands correctly. This host CPU cycles can be claimed even if the host CPU to do nothing. To avoid this problem, the tool should *dosidle* ([ftp://ftp.volftp.tiscali.it/pub/pc/msdos/utility/dosidle210.zip](http://ftp.volftp.tiscali.it/pub/pc/msdos/utility/dosidle210.zip)) system to be installed in the guest. Due to a bug in QEMU, it occurs in some mouse and keyboard drivers for DOS problems. *The-rtc base = localtime* is DOS and Windows for the time required for proper system. For problems with time synchronization is Tardis (<http://www.kaska.demon.co.uk>) to install, time synchronization with a time server is managing the. Free application software is among the URLs <http://www.webi.org> , <https://ninite.com> , <http://portableapps.com/de/apps> , <http://www.lupopensuite.com> , <http://www.freewaregeeks.com> and <http://www.opensource-dvd.de/isodownload.htm> to find. Web designers can from the URL <http://spoon.net/browsers/> browser as executable files more. These browsers run without installation. Images of boot disks can be found at <http://www.bootdisk.com> .

CP/M-86

Download: <http://www.gaby.de/ftp/pub/cpm/sysdisks/cpm86/86raw144.zip>
Recommendation: Standard PC (standard), 2 MB RAM, hard disk: up to 8 MB.

The operating system CP / M (Control Program for Microcomputers) was developed in 1974 by Digital Research Inc.. During this time, the memory is specified in Kbytes, floppy drives were not self-evident and hard drives almost prohibitive. CP / M was the first platform-independent operating system for 8-bit machine. only the lowest level of CP / M, the Basic Input Output System (BIOS) had adapted to the hardware. The upper layers (BDOS - Basic Disk Operating System and CCP - console command processor) could be ported without major changes. With the start of the 16-bit era CP / M was on 8086 - and 8088-ported CPUs from Intel. This port was called CP / M 86. As a standard operating system for the IBM PC could not CP/M-86 prevail. Microsoft MS-DOS made it to market better. In 1988, a highly evolved, fully MS-DOS-compatible version of CP / M 86 released as DR-DOS and Microsoft once again made a number of years serious competition. By the way, are used in Windows inflexible addressing of drives and hard disk partitions with drive letters not superseded heritage of CP / M. The operation of a floppy disk CP/M-86 enough. The disk image is to download and unpack. Next, the system is started from the image. It is necessary to use *the-no-fd-bootchk* (see http://qemu-buch.de/d/Speichermedien/Zugriff_auf_Speichermedien).

```
Host ~ $ wget \
  http://www.gaby.de/ftp/pub/cpm/sysdisks/cpm86/86raw144.zip
Host ~ $ unzip 86raw144.zip
Host ~ $ qemu-fda 144cpm86.img-no-fd-bootchk-boot a \
  Rtc-base = localtime-m 2-vga cirrus
```

CP/M-86 can address a hard disk with up to 8 MB.

```
Host ~ $ qemu-img create-f qcow2 cpm86.img 8M
Host ~ $ qemu-fda 144cpm86.img-no-fd-bootchk-boot a \
  Rtc-base = localtime-m 2-vga cirrus isapc-no-acpi \
  -Drive file = cpm86.img, if = ide, index = 0, media = disk
```

In CP/M-86 *HDMAINT.CMD* used the program for partitioning hard drives. The operation of *HDMAINT.CMD* done with the function keys.



Microsoft Windows XP as a guest system under Q (Mac OS X).



Microsoft Home Server as a guest system under QEMU.



Microsoft Windows Vista in the Kernel-based Virtual Machine.



Microsoft Windows 7 as a guest system under QEMU / KVM.



ReactOS under QEMU.

Guest A> **hdmaint [F1] Display or change disk partitioning [F6]** Change or create CP / M partition Enter starting cylinder:

For reading the partition table part of the system is restarted. The drive is shipped with *B:* addressed. Files to copy using the tool below CP/M-86 *PIP.CMD*.

```
Guest A> pip b: = a: *.*  
A guest> dir b:
```

DR-DOS 7.03

Download: <http://www.drdos.net/download.htm>

Recommendation: Standard PC (default), 16 MB RAM, hard disk: 0.5 GB.

DR-DOS is compatible with MS-DOS operating system. It was developed by Digital Research CP / M 86 from their operating system. Because of its qualities and its stability, DR-DOS was already after its release as an equal competitor to MS-DOS. The installation requires a virtual hard disk.

```
Host ~ $ qemu-img create-f qcows drdos.img 512M
```

To download from five floppy images available (*disk01.144* to *disk05.144*). After downloading, you start the virtual machine.

```
Host ~ $ qemu-hda drdos.img disk01.144-fda-boot a \  
Rtc-base = localtime-m 16
```

To replace the floppy with the key combination [Ctrl] + Switch to [Alt] + [2] in the QEMU monitor. In this example, the floppy drive, the name *floppy0*.

```
(Qemu) change floppy0 disk02.144
```

The host system brings you back with [Ctrl] + [Alt] + [1]. The restart is done with this command:

```
Host ~ $ qemu-rtc drdos.img base = localtime-m 16
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b drdos.img qcows DRDOS-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-DRDOS 01.ovl-rtc base = localtime-m 16
```

FreeDOS 1.0

Download: <http://www.freedos.org/freedos/files/> (*fdfullcd.iso*)

Recommendation: Standard PC (default), 16 MB RAM, hard disk: 0.5 GB.

As Microsoft had discontinued the development of MS-DOS, developed the project FreeDOS. The software is very advanced and provides functions and features that MS-DOS can miss. At the beginning of the installation is the creation of a virtual disk.

```
Host ~ $ qemu-img create-f qcows freedos.img 500M
```

You start the virtual machine with the following options.

```
Host ~ $ qemu-hda-cdrom freedos.img fdfullcd.iso-boot d \  
16-m-rtc base = localtime
```

When you start the mouse focus should remain in the window of the instance, otherwise the mouse is not detected properly. You select the option to install to the hard disk. Then asked to select the language and keyboard layout. In the next steps the hard disk is partitioned with *XFDisk*. It creates a primary partition with the overall size of the hard disk. Then install the boot manager. Required is an entry in the boot menu, for example, "FreeDOS". With [F3], leave *XFDisk* and the instance is to reboot. After the reboot, we again chose the option to install on your hard drive. After formatting the hard drive is started with the installation of FreeDOS. The installer of FreeDOS has a simple package management system, which is called automatically after formatting the disk. As a network driver should QEMU-compatible drivers are selected. After installation, network configuration is by modifying the *C: FDOS \ BIN \ Wattcp.cfg*.

```
my_ip = dhcp  
netmask = 255.255.255.0  
gateway = 0.0.0.0  
domain_list = your.domain.com
```

In order for the network under FreeDOS, QEMU or KVM is to start with the following options:

```
Host ~ $ qemu-m 16-freedos.img rtc base = localtime \  
-Net user, hostname = freedos-vm-net nic, model = ne2k_isa
```

For network support, the network card *NE2000* ISA bus (*ne2k_isa*) are required. Furthermore, a host name, in this *freedos-vm* configured. Due to network support, the Arachne web browser can be used.

```
Guest C: \> arachne
```

For there is the FreeDOS GUI OpenGEN that is installed via package management. The start is the command *in accordance* with OpenGEN.

```
Guest C: \> gem
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b freedos.img qcow2 freedos-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu freedos-01.ovl-m 16-rtc base = localtime \
-Net user, hostname = freedos-vm-net nic, model = ne2k_isa
```

DESKWORK

Download: <http://www.deskwork.de/DOWNLOAD>

Recommendation: Standard PC (default), from 8 MB RAM, hard disk: 0.5 GB.

DESKWORK is an extension to DOS. It supports long file names and meta information. Files can be filtered by type of content. Directories are not available. Go to a DOS installation is required, in this example it is FreeDOS ODIN 0.7. A virtual disk is under <http://www.nongnu.org/qemu/download.html> downloaded.

```
Host ~ $ wget http://odin.fdos.org/odin2005/odin1440.img
```

With *qemu-img* create a virtual hard drive.

```
Host ~ $ qemu-img create-f qcow2 DESKWORK.img 500M
```

You start the virtual machine with the following options.

```
Host ~ $ qemu-fda odin1440.img DESKWORK.img-hda-boot a-no-kvm
```

In the host system, the virtual disk is partitioned.

```
Guest A: \> fdisk
Do you want to use large disk (FAT32) support (Y / N). [Y]?
1st Create DOS partition or Logical DOS Drive
1st Create Primary DOS Partition
Do you wish to use the maximum available size for a
Primary DOS Partition and make the partition active (Y / N)? [Y]
[ESC]
[ESC]
```

Then a restart is required. Then, the virtual hard disk in the host system is formatted.

```
Guest A: \> format c:
```

The virtual machine is shut down again and it loads the DESKWORK files from the site in a subdirectory of the host system.

```
Host $ mkdir ~ DESKWORK inst
Host $ cd ~ DESKWORK inst
Host ~ $ wget http://www.deskwork.de/DOWNLOAD/CURRENT/APPS.DW
Host ~ $ wget http://www.deskwork.de/DOWNLOAD/CURRENT/MAIN.DW
Host ~ $ wget http://www.deskwork.de/DOWNLOAD/CURRENT/PATCHES.DW
Host ~ $ wget http://www.deskwork.de/DOWNLOAD/CURRENT/SETUP.EXE
Host ~ $ wget http://www.deskwork.de/DOWNLOAD/CURRENT/SYSTEM.DW
Host ~ $ wget http://www.deskwork.de/DOWNLOAD/CURRENT/HARDWARE.TXT
Host $ cd ~ -
```

This directory is mounted as a virtual FAT disk.

```
Host ~ $ qemu-fda-hda odin1440.img DESKWORK.img \
-Hdb fat: DESKWORK-inst-boot a-no-kvm
```

In the guest system is changed to the second virtual hard drive and start the setup.

```
Guest A: \> d:
Guest D: \> setup
```

After installing DESKWORK starts.

```
Guest C: \ DW> dw
```

The options allow for better hardware emulation for DESKWORK.

```
Host ~ $ qemu-fda odin1440.img DESKWORK.img-hda-boot a \
Std-vga-soundhw es1370-rtc base = localtime-no-kvm
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b DESKWORK.img qcow2 DESKWORK-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-fda-hda odin1440.img DESKWORK-01.ovl-boot-a \
Std-vga-soundhw es1370-rtc base = localtime-no-kvm
```

OS / 2 Warp 4

Recommendation: Standard PC (default), 64 MB RAM, hard disk: 1 GB.

OS / 2 is a multitasking-capable, stable operating system for PCs. It was originally jointly developed as a successor to MS-DOS from IBM and

Microsoft. After Microsoft withdrew in 1991, IBM developed it further. OS / 2 can run alongside the specially developed programs and MS-DOS programs and Windows 3.1 programs. OS / 2 is still occasionally in banks, insurance companies and airlines to be found. It is used by IBM or not formally developed. New installations are usually the eComStation distribution (<http://www.ecomstation.de> realized). IBM advises customers to switch to Linux. The virtualization of OS / 2 is problematic, since this system uses rarely used commands of the processor. To install a virtual machine first, an image of the installation CD is to be generated. Furthermore, the image of the boot disks are required. These are located in *diskimgs/os2/35* on the installation CD. The floppy images are *disk2.dsk* and *disk0.dsk disk1_cd.dsk* hard disk of the host system to copy the. With *qemu-img* create a virtual hard drive.

```
Host ~ $ qemu-img create-f qcows os2w4.img 1G
```

The virtual disk is initially under DOS partition and format. This example is used FreeDOS ODIN 0.7. This is the virtual disk from <http://www.nongnu.org/qemu/download.html> downloaded.

```
Host ~ $ wget http://odin.fdos.org/odin2005/odin1440.img
```

You start the virtual machine with the following options.

```
Host ~ $ qemu-fda odin1440.img os2w4.img-hda-boot a-no-kvm
```

In the host system, the virtual disk is partitioned.

```
Guest A: \> fdisk
Do you want to use large disk (FAT32) support (Y / N). [Y]?
1st Create DOS partition or Logical DOS Drive
1st Create Primary DOS Partition
Do you wish to use the maximum available size for a
Primary DOS Partition and make the partition active (Y / N)? [Y]
[ESC]
[ESC]
```

A reboot is necessary. Then, the virtual hard disk in the host system is formatted.

```
Guest A: \> format c:
```

You start the virtual machine and performs the installation according to the instructions in the manual.

```
Host ~ $ qemu-hda-fda os2w4.img disk0.dsk \
Cdrom-os2-warp4.iso-boot a-m 64-rtc base = localtime-no-kvm
```

Since OS / 2 Warp 4 a bit older, legacy hardware must be emulated. As the virtualization of OS / 2 Warp is problematic, the *option-no-kvm* apply. To replace the floppy with the key combination [Ctrl] + Switch to [Alt] + [2] in the QEMU monitor. In this example, the floppy drive, the name *floppy0*.

```
(Qemu) change floppy0 disk1_cd.dsk
```

With the key combination [Ctrl] + [Alt] + [1] you go back. The third disk is replaced with the following command:

```
(Qemu) change floppy0 disk2.dsk
```

After the restart following options are entered:

```
Host ~ $ qemu-m 64-rtc os2w4.img base = localtime-no-kvm
```

Notes on configuration can be found in the manuals of OS / 2 Warp 4th It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b os2w4.img qcows os2w4-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-m 64 01.ovl os2w4-rtc-base = localtime-no-kvm
```

Microsoft Windows (for Workgroups) 3.xx

Recommendation: Standard PC (default), 16 MB RAM, hard disk: 0.5 GB.

Microsoft Windows 3.x is not a separate operating system. It is an extension to DOS to some of its drawbacks (poor memory management, lack of GUI) to minimize. Microsoft Windows for Workgroups supports network functionality. From Microsoft Windows for Workgroups 3.1 and with the package *winsocks* protocol is to use the TCP / IP is possible. Here is an example of the described installation of Windows 3.1 with DR-DOS. One to copy the image with DR-DOS (example Unix-/Linux-Befehlen).

```
Host ~ cp ~ drdos.img win31.img
```

Microsoft Windows 3.1 was shipped on multiple discs. floppy disk as an image not import any order must name the contents of the floppy disk in a directory with the copied *win31inst*. This directory is mounted as a virtual FAT disk.

```
Host ~ $ qemu-hda-hdb fat win31.img: win31inst \
Rtc-base = localtime-m 16-soundhw sb16-no-kvm
```

In the guest system you switch to the *D:* drive and run the setup program.

```
Guest C: \> d:
Guest D: \> setup
```

Further details of the installation can be viewed in the Windows 3.x manuals. After installation, you start the virtual machine with the following options:

```
Host ~ $ qemu-rtc win31.img base = localtime-m 16-soundhw sb16
```

Tips and software for Windows 3.x can be found at the URLs <http://www.win31.de> and <http://wiki.oldos.org>. Those who prefer a more modern GUI for Windows 3.x will use an install Calmira (<http://www.calmira.de>). It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b win31.img qcow2 win31-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-win31-rtc 01.ovl base = localtime-m 16-soundhw sb16
```

Microsoft Windows 95

Recommendation: Standard PC (default), 16 MB RAM, hard disk: 0.5 to 2 GB.

Microsoft Windows 95 is a system based on MS-DOS operating system. In terms of stability, it is not sufficient approach to Windows 3.11 or Windows versions based on NT. Windows 95 was either delivered on multiple disks or on an installation CD with a boot disk. At the time PCs were those that could boot from CD-ROM, not a given. The following example describes an installation boot diskette and CD. First imported to the installation CD and boot floppy. Furthermore, I create a virtual hard disk.

```
Host ~ $ qemu-img create-f qcow2 win95.img 500M
```

You start the instance with the following options:

```
Host ~ $ qemu-hda win95-fda-startfloppy.img win95 \
  Cdrrom-win95-install-cd.iso-boot a-no-acpi-isapc M \
  16-m-rtc-base = localtime soundhw sb16-no-kvm \
  -Net user, hostname = win95-vm-net nic, model = ne2k_isa
```

You start the virtual machine and performs the installation according to the instructions in the manual. After installation, the following options apply:

```
Host ~ $ qemu-M win95.img isapc-no-acpi-m 16 \
  -Net user, hostname = pc-qemu-net nic, model = ne2k_isa \
  Rtc-base = localtime-soundhw sb16 \
```

Installing the latest service pack is recommended. You find it under <http://www.treiber-forum.de/servicepacks/>. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b win95.img qcow2 win95-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-win95-M 01.ovl isapc-no-acpi-m 16 \
  -Net user, hostname = pc-qemu-net nic, model = ne2k_isa \
  Soundhw-rtc-sb16 base = localtime
```

Microsoft Windows 98

Recommendation: Standard PC (default), 64 MB RAM, hard disk: 1 GB.

Microsoft Windows 98 is based on MS-DOS 7.1. Innovations were the better support of AGP (Accelerated Graphics Port) and USB and ACPI. Windows 98 SE (Second Edition) contains, among other things, DirectX 6.1, Internet Explorer 5.0, Windows Media Player 6.1 and better networking support. First imported to the installation CD as an image and creates a virtual hard disk.

```
Host ~ $ qemu-img create-f qcow2 win98.img 1G
```

You start the virtual machine and performs the installation according to the instructions in the manual.

```
Host ~ $ qemu-hda-cdrom win98.img win98inst.iso-boot d \
  -No-acpi-m 64-rtc base = localtime-usb-soundhw sb16-no-kvm
```

When the installation is finished, the instance is started with the following options:

```
Host ~ $ qemu-no-acpi win98.img-m 64-rtc base = localtime-usb-soundhw sb16
```

The installation of the service pack is recommended. They are found under <ftp://ftp.dfn-cert.de/pub/vendor/microsoft/win98/> and <http://www.treiber-forum.de/servicepacks/>. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b win98.img qcow2 win98-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-win98 01.ovl-no-acpi-m 64-rtc base = localtime-usb \
  Soundhw-sb16
```

Microsoft Windows NT

Recommendation: Standard PC (default), 32 MB RAM, Hard Drive: 2 GB

Microsoft Windows NT is completely detached from MS-DOS. The DOS prompt is only available as an emulator. It also supports Windows NT file system NTFS (New Technology File System). NT was originally designed for different processor architectures and dominated preemptive multitasking with memory protection. Direct hardware access programs are not covered by the strict enforcement of a layered model allows. From NT 4.0 does the GDI graphics subsystem performance reasons in part directly in the operating system kernel, which video driver bug in NT versions may crash. To install the installation CD is imported as an image. Furthermore, I create a virtual hard disk.

```
Host ~ $ qemu-img create-f qcow2 win nt.img 2G
```

You start the virtual machine and perform the installation as derAnleitung in the manual.

```
Host ~ $ qemu-hda-cdrom win nt.img winNTw-insCD.iso-boot d \
Rtc-base = localtime-m 32-soundhw sb16-no-kvm
```

When the installation is finished, the instance is started with the following options:

```
Host ~ $ qemu-win nt.img-rtc base = localtime-m 32-soundhw sb16
```

The installation of the service pack is recommended. They are found under <ftp://ftp.dfn-cert.de/pub/vendor/microsoft/> and <http://www.treiber-forum.de/servicepacks/>. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b-f nt.img win-win-nt-qcow2 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-win-nt-rtc 01.ovl base = localtime-m 32-soundhw sb16
```

Microsoft Windows 2000

Recommendation: Standard PC (default), 128 MB RAM, hard disk: 5 GB.

Microsoft Windows 2000 (NT 5.0), a technical development of Microsoft Windows NT 4.0, dar. New in this version, including the Active Directory. In addition, access other open standards, such as the MIT developed Kerberos to authenticate and authorize users and network services. USB mass storage devices are supported without driver installation. All versions of Windows from Microsoft Windows 2000 install different kernel versions, depending on whether ACPI is available. Therefore should not be disabled later in QEMU or KVM ACPI, when it was enabled during installation. When you install Windows 2000, the *option-win2k-hack* needed because a bug in Microsoft Windows 2000, the hard drive when you install a fully characterized by a will. Once installed, this option is not necessary. To install the distribution media is imported as images. Furthermore, I create a virtual hard disk.

```
Host ~ $ qemu-img create-f qcow2 win00.img 5G
```

You start the virtual machine and performs the installation according to the instructions in the manual.

```
Host ~ $ qemu-hda win00.img-fda-win00-rtc dsk01.img base = localtime \
Win00.iso-cdrom-boot a-soundhw all-no-kvm-win2k-hack
```

To replace the floppy with the key combination [Ctrl] + Switch to [Alt] + [2] in the QEMU monitor. In this example, the floppy drive, the name *floppy0*.

```
(Qemu) change floppy0 win00-dsk02.img
```

For the emulator window you go with [Ctrl] + [Alt] + [1] back. When the installation is finished, Windows will start 2000 with the following options:

```
Host ~ $ qemu-rtc win00.img base = localtime-soundhw all
```

The installation of the service pack is recommended. They are found under <ftp://ftp.dfn-cert.de/pub/vendor/microsoft/> and <http://www.treiber-forum.de/servicepacks/>. Microsoft Windows 2000 does not close automatically after shutting down the instance. The cause lies in the non-activated APM driver. Comprising the steps of the APM driver is installed.

```
Control Panel, Hardware,
Add / Troubleshoot a device,
Add a new device, further
No, the hardware components to choose from the list
NT APM / Legacy support, Next, Next, Finish
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b win00.img qcow2 win00-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-win00-rtc 01.ovl base = localtime-soundhw all
```

Microsoft Windows XP

Recommendation: Standard PC (default), 256 MB RAM, hard disk: 5 GB.

Microsoft Windows XP (NT 5.1) is the successor to Windows 2000. Microsoft Windows XP, there is, among other things in these versions: The Professional Edition for enterprise use, the Home Edition as a low cost option and the very limited Starter Edition. Unlike in previous versions of Windows NT Server, there is no version of XP. All versions of Windows 2000 and install different kernel versions, depending on whether ACPI is available. Therefore should not be disabled later in QEMU or KVM ACPI, when it was enabled during installation. To install the distribution media is imported as images and creates a virtual hard disk.

```
Host ~ $ qemu-img create-f qcow2 win xp.img 10G
```

You start the virtual machine and performs the installation according to the instructions in the manual.

```
Host ~ $ qemu-hda win-win-xp xp.img-cdrom-boot d-cd.iso \
-M 256-localtime-rtc base = soundhw es1370-vga std-no-kvm
```

When the installation is complete, the following options are necessary.

```
Host ~ $ qemu-win xp.img-m 256-std-vga soundhw es1370
```

It is recommended that you install Service Pack 3. It is also necessary for the operation of a current anti-virus program. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b-f xp.img win-win-xp-qcow2 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-win-xp-m 256-01.ovl soundhw es1370 \
Std-vga-rtc base = localtime-usb-tablet USBDevice
```

In KVM following options are applicable.

```
Host ~ $ kvm-win xp.img-m 256-std-vga soundhw es1370-rtc-base = localtime \
-Net nic, model = rtl8139-net user-usb-tablet USBDevice
```

It is possible, via Remote Desktop Protocol (see # http://qemu-buch.de/d/Anhang/_Nützliche_Tools_Remote/Desktop_Protocol on the host system log).

Microsoft Windows Home Server

Download: <http://www.microsoft.com/downloads/details.aspx?FamilyID=0960c088-c539-4265-9c5c-84fb6b2dadca&displaylang=en> (test version for 120 days)

Recommendation: Standard PC (default), 512 MB RAM, Hard Drive: 70 GB.

The Microsoft Windows Home Server is based on the Microsoft Windows Server 2003 (Small Business Edition) and is designed for the home network. The actual home server is a tower. First, a virtual hard disk must be created.

```
Host ~ $ qemu-img create-f qcow2 WindowsHomeServer.img 70G
```

Then QEMU or KVM starts. Carry out the installation by following the instructions in the manual.

```
Host ~ $ qemu-hda WindowsHomeServer.img \
-Cdrom Windows \ Home \ Server \ DVD.iso \
-Boot d-m 512-rtc base = localtime-usb-vga std \
-Base = 2009-01-12 rtc-no-kvm
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b WindowsHomeServer.img \
-F qcow2 windows home server 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu windows home server 01.ovl-m 512-rtc base = localtime-usb \
Std-vga-usb-tablet-rtc USBDevice base = 2009-01-12
```

Microsoft Windows Vista

Recommendation: Standard PC (default), 2048 MB RAM, hard disk: 20 GB.

Microsoft Windows Vista (NT 6.0) is the successor of the operating system Microsoft Windows XP. Microsoft Windows Vista is available on DVD in six versions: Starter, Home Basic, Home Premium, Business, Enterprise and Ultimate. The versions differ in functionality and price. Microsoft Windows Vista will not boot on older versions of QEMU and KVM. First, a virtual hard disk must be created.

```
Host ~ $ qemu-img create-f qcow2 win vista.img 20G
```

Then QEMU or KVM starts. Carry out the installation by following the instructions in the manual.

```
Host ~ $ qemu-hda-cdrom win vista.img Windows_Vista_RC1_32Bit.iso \
-M 1024-smp 2-base = 2006-01-01 rtc-boot d-no-kvm
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b-f vista.img win-win-vista-qcow2 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-win-vista 01.ovl-m 1024-smp 2-net nic, model = e1000 \-net user-usb-tablet-rtc USBDevice base = 2006-01-01
```

It is possible, via Remote Desktop Protocol (see # http://qemu-buch.de/d/Anhang/_Nützliche_Tools_Remote/Desktop_Protocol on the host system log).

Microsoft Windows 7 Enterprise (32 -, 64-bit)

Download: <http://technet.microsoft.com/de-de/evalcenter/cc442495.aspx?ITPID=wtcfeed> (90-day trial)

Recommendation: Standard PC 64-bit, 1024 MB RAM, Hard Drive: 16 GB.

Microsoft Windows 7, the successor to Windows Vista. Development takes place on a common code base with Windows Server 2008 R2. It is a virtual hard drive to create.

```
Host ~ $ qemu-img create-f qcow2 win7.img 20G
```

For the 32-bit version of QEMU or KVM starts with the following options. For problems with older versions of QEMU verwednet one additional option-no-kvm.

```
Host ~ $ qemu-hda-cdrom win7.img windows7.iso-m 1024-boot d \
USBDevice-usb-tablet-rtc base = 2009-05-05
```

For the 64-bit version of the following options are applicable. The *option-cpu* *qemu64*, + *svm* is only necessary if the host system on AMD processors, and a nested virtualization is desired.

```
Host ~ $ qemu-system-x86_64-cpu qemu64, + svm-enable-kvm-hda win7.img \
-Cdrom-7000.0.081212 1400_client_de-de_Ultimate-GB1CULFRE_DE_DVD.iso \
D-m 1024-boot-usb-tablet-rtc USBDevice base = 2009-05-05
```

Since the system time host system is synchronized with the time server time synchronization via Control Panel, *date and time, time* to turn off the *Internet* in. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b win7.img qcow2 win7-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-m 01.ovl win7-1024-std-vga usb USBDevice tablet \-rtc base = 2009-05-05
```

For the 64-bit version of the following options are applicable.

```
Host ~ $ qemu-system-x86_64-cpu qemu64, + svm-enable-kvm-hda-win7 01.ovl \
D-m 1024-boot-usb-tablet-rtc USBDevice base = 2009-05-05
```

Windows 7 is running without having to enter the license key for 30 days. This period can use the following command with administrative privileges extend up to three times.

```
Guest C: \> slmgr-rearm
```

ReactOS 0.3.12

Download: <http://www.reactos.org/de/download.html>

Recommendation: Standard PC (default), 512 MB RAM, hard disk: 5 GB.

ReactOS should be compatible with the kernel of Microsoft Windows NT. This makes it possible to use programs and drivers for NT and its successor, 2000, XP, Vista and 7. This is working on the reproduction of the programming interface (API), Win32, Win16, OS / 2, Java and DOS. The operating system is released under the GPL. Thus, it would be possible to get independent from Microsoft and free alternative to Windows. The project is currently in alpha stage and so ReactOS is not recommended for everyday use. To install the operating system a virtual disk is created.

```
Host ~ $ qemu-img create-f qcow2 ReactOS.img 5G
```

From the ReactOS website is the image of the installation CD to download and unpack.

```
Host ~ $ unzip ReactOS *- iso.zip
```

QEMU / KVM is started with the following options.

```
Host ~ $ qemu-hda-cdrom ReactOS.img ReactOS.iso-boot d \
-M 512-net user-net nic, model = PCnet
```

Installing ReactOS is started by clicking in the QEMU window and pressing any key. First, the German keyboard is selected. Thereafter, the virtual disk is partitioned and formatted. The target directory for the installation of the ReactOS files need not be changed. After the boot loader is installed, the default option *Install bootloader on the harddisk (MBR) holds*. Then the virtual machine must be restarted. The installation continues graphically.

```
Host ~ $ qemu-net user-net ReactOS.img nic, model = PCnet \
M-512-rtc base = localtime
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b ReactOS.img qcow2 ReactOS 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-net user ReactOS 01.ovl-net nic, model = PCnet \
M-512-rtc base = localtime
```

To compile ReactOS is the *ReactOS Build Environment* (see http://qemu-buch.de/d/Anhang/_Diverses/_RosBE).

<<< | # # # | >>> <http://qemu-buch.de>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_x86-Architektur/_DOS-%_2C_Windows_und_Verwandte "

This page has been accessed 16,046 times. This page was last updated on 27 January 2011 at 21:13 clock changed. Content is available under GNU Free Documentation License 1.2 .

BSD Unix systems, Solaris 10 OpenSolaris Illumos, FreeBSD 8.0 manual, PC-BSD 8.1, OpenBSD 4.8, NetBSD 5.0, Nexenta OS, Darwin, download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / x86 / BSD Unix systems]])

<<< | # # # | >>> | English

Contents

- 1 BSD-Unix systems
 - 1.1 FreeBSD 8.0
 - 1.2 PC-BSD 1.8 Hubble Edition
 - 1.3 OpenBSD 4.8
 - 4.1 NetBSD 5.0.2
 - 4.1.1 Installation CD Image
 - 4.1.2 Mini Appliance
 - Solaris 10 01/05 - 10/08 (Intel), OpenSolaris, Illumos
Nexenta 1.6.2.0
Darwin 1.7.8.01 (Intel)

BSD Unix systems

Unix systems are multi-user operating systems. Unix has been programmed in the early 70s of the twentieth century by Bell Laboratories in support of software development. Unix called in common language, operating systems that either originated in the Unix system, where AT & T (formerly Bell Laboratories) or have implemented the concepts. Because Unix is a registered trademark of The Open Group, only certified systems use the name Unix. For all these systems, which can be divided into Unix systems and Unix-like operating systems include, for example, the BSD systems, Mac OS X, HP-UX, AIX, IRIX and Solaris. Some other systems such as GNU Linux or QNX are in the historic sense, no Unix derivatives. BSD was originally based on Bell Labs source code, but these were completely removed by mid-90s. A "hit parade" of the BSD systems located at the URL <http://www.bsdstats.org>.

FreeBSD 8.0

Website: <http://www.freebsd.org/de>

Recommendation: Standard PC, 256 MB RAM, 10 GB hard drive.

FreeBSD is a free Unix-derivative of the BSD family, focusing on the x86 architecture. FreeBSD is popular among Internet providers, because it is robust. Parts of the source code of FreeBSD are also used in various proprietary operating systems such as Mac OS X,. This supports the BSD license. There are several FreeBSD distributions, such as PC-BSD. FreeBSD Handbook can be found at the URL http://www.freebsd.org/doc/de_DE.ISO8859-1/books/handbook/. For a lean server installation wealthy few GB of hard drive capacity.

```
Host ~ $ qemu-img create-f qcows freebsd.img 10G
```

Further invites the image file you downloaded from the site and start the virtual machine.

```
Host ~ $ wget \ ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.0/8.0-RELEASE-i386-bootonly.iso host ~ $ qemu-hda-m 256 freebsd.img -boot
```

The installation is done in text mode. First, the land and the keyboard layout is selected. Then drive in the breakdown of the slices with *fdisk*. A (Use Entire Disk) you use the entire virtual hard disk. It marked the slice *ad0s1* and press the *S* key to a boot-able to define this slice. This is an *A* in the column marked with *flags*. To exit *fdisk*, press the *Q* key. Then the boot manager is configured. This *manager* is the *default boot* selected. On FreeBSD two layers are used in the partitioning. The lower layer is similar to the partitioning in other systems and is visible to those systems. This step has already been processed with *fdisk*. The second layer is divided into the first layer further individual file systems. To this end, the tool *disk label* with the FreeBSD partition divided. To make the distribution of partitions automatically, *A* is the key to press. Then we ended the program with *Q*.

The next step is to select the distribution. installed depending on the distribution and intended use (server, desktop, ...) are different software packages. Is chosen with *all* of everything. To install software with ports collection is the *FreeBSD ports* to install. Because the software is downloaded over the Internet, the download is configured. One chooses a *passive* protocol and a mirror server from *FTP*. The network settings are configured via DHCP. After the system was installed, the virtual machine is restarted. The configuration of the system is the command *sysinstall* with. This means, for example, network services and set up User. The command *shutdown-h now halt* command or the system, the shut down. The virtual machine is started.

```
Host ~ $ qemu-m 256 freebsd.img
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b freebsd.img qcows freebsd-hd-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu freebsd-hd-01.ovl-m 256
```

In KVM options are possible:

```
Host ~ $ kvm freebsd-hd-01.ovl-m \ 256
-Net user-net nic, model = ne2k_pci
```

PC-BSD 8.1 Edition

Website: <http://www.pcbsd.org>

German Portal: <http://www.pcbsd.de>

Recommendation: Standard PC, 256 MB RAM, 10 GB hard drive.

PC-BSD is a free, open-source operating system based on FreeBSD. The project has set itself the goal of an easy to install and to develop easy-to-use operating system for end users. PC-BSD uses a graphical installer and a graphical software installation program with PC-BSD Packages (.Pbi). Parallel continue the ports system of FreeBSD is available. To install I create a virtual hard disk.

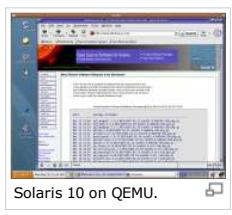
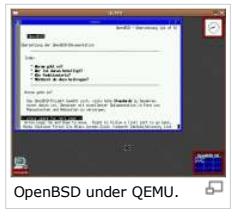
```
Host ~ $ qemu-img create-f qcows pcbsd.img 10G
```

Now you start the virtual machine.

```
Host ~ $ qemu-hda-cdrom pcbsd.img PCBSD8.1 bootonly.iso x86 \
-Boot d-m 256-usb-soundhw all
```

PC-BSD 8.1 has a new installer that can be used to either FreeBSD and PC-BSD install. First, the language, keyboard and time zone are selected. After that, the installation version, desktop or server, select it. Furthermore, the installation method to *install from the network* set. Your networking setting is queried. Here are the specifications (RealtTek 802.11, DHCP) are to take over. For the server installation, the default setting is confirmed. In step *system accounts* is the root password is set and leave it in or create multiple user accounts. In the *partitioning* step selects from the entire hard disk and can install the bootloader. Then the software packages will be selected. After Instatllation a restart is necessary:

```
Host ~ $ qemu-hd.img PCBSD-m 256-localtime-rtc base = soundhw all
```



It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b PCBSD-hd.img qcow2 PCBSD-f-hd-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu PCBSD-hd-01.ovl-m 256-localtime-rtc base = soundhw all
```

OpenBSD 4.8

Download: <http://www.openbsd.org/de/> (*cd48.iso*)

Recommendation: Standard PC, 128 MB RAM, minimum 2 GB hard disk.

OpenBSD is a 4.4BSD-based operating system that is freely available from the OpenBSD license. OpenBSD includes a number of security features available in other operating systems are not or only option available. Because of cryptography, security improvements, and the integrated firewall in OpenBSD is particularly suitable for use as a firewall, intrusion detection system and VPN gateway. It is also often used for secure Web and other servers. OpenBSD has some quirks that even many experts are foreign to Unix. For a lean server installation wealthy few GB of hard drive capacity.

```
Host ~ $ qemu-img create-f qcow2 openbsd.img 2G
```

It will download to the image file.

```
Host ~ $ qemu-hda openbsd.img-cdrom-boot d cd48.iso
```

The installation is done in text mode. Default values in square brackets can be taken over by [Enter].

```
(I) nstall, (U) pgrade or (S) light is I  
Specify terminal type: [vt220] [Enter]
```

For the selection of the keyboard layout is *de* or *none* (U.S. keyboard layout) specified.

```
kbd (8) mapping? ('L' for list) [none] com  
Proceed with install? [No] y
```

IDE hard drives with OpenBSD as *wd0*, *wd1* and further labeled as such.

```
Available disks are: wd0.  
Which one is the root disk? (Or done) [wd0] [Enter]
```

It makes use of the virtual hard disk for OpenBSD, yielding the partitioning with *fdisk* spared.

```
Do you want to use * all * of wd0 for OpenBSD? [No] y
```

In OpenBSD, two layers are used in the partitioning. The lower layer is similar to the partitioning in other systems and is visible to those systems. The second layer is divided into the first layer further individual file systems. It has indeed saved *fdisk*, *disklabel* tool but must use the.

```
Initial label editor (enter '?' For help at any prompt)  
>?
```

On the root disk partitions, at least the *a* and *b* are generated. For the root file system (/) here is *a* and *b* used for swap. The swap partition is replaced by the 300 MB and the rest goes to the root file system. First, can you look at the labels (*p*) in MB (*m*) show.

```
> Pm
```

The *c* partition indicates the way the entire hard disk and can not be changed. It only deletes *a* partition.

```
> Da
```

You put a partition in the desired size new.

```
> Aa  
offset: [63] [Enter]  
size: [20964762] 20349945  
FS type: [4.2BSD] [Enter]  
mount point: [none] /
```

You create the swap partition.

```
> Fm  
offset: [20350008] [Enter]  
size: [614 817] [Enter]  
FS type: swap
```

To check you can view the labels.

```
> Pm
```

You end the program *disk label*.

```
> Q  
Write new label: [y] [Enter]  
The next step * DESTROYS * all existing data on these partitions!  
Are you really sure that you're ready to proceed? y
```

The installation continues and it is the host name set.

```
System hostname? (Short form, eg 'foo') foo (or reasonable)
```

The network is configured.

```
Configure the network? [Yes] [Enter]  
Which one do you wish to initialize? (Or 'done') [NE3] [Enter]  
Symbolic (host) name for ne3? [Foo] [Enter]  
Do you want to change the default media? [No] [Enter]  
IPv4 address for ne3 (or 'none' or 'dhcp')? Dhcp  
IPv6 address for fxp0? (Or 'rtsol' or 'none') [none] [Enter]  
DNS domain name? (Eg 'bar.com') [my.domain] example.com  
DNS name server? (IP address or 'none') [10.0.2.3] [Enter]  
Use the nameserver now? [Yes] [Enter]  
Default route? (IP address, 'dhcp' or 'none') [dhcp] [Enter]  
Edit hosts with ed? [No] [Enter]  
Do you want to do any manual network configuration? [No] [Enter]
```

It sets the password for the *root* user.

```
Password for root account? (Will not echo) *****
```

Password for root account? (Again) *****

To install the software packages installation media must be provided.

```
Location of sets? (Cd disk ftp http or 'done') [cd] [Enter]
Which one contains the install media? (Or 'done') [cd0] [Enter]
Pathname to the sets? (Or 'done') [4.1/i386] [Enter]
```

Then the sets are selected. Sets are groups of software packages.

```
[X] bsd.rd
Set name? (Or 'done') [done] [Enter]
Ready to install sets? [Yes] [Enter]
Location of sets? (Cd disk ftp http or 'done') [cd] [Enter]
```

The CD image does not contain all necessary components. The rest will be downloaded over the Internet.

```
Location of sets? (Cd disk ftp http or 'done') ftp
HTTP / FTP proxy URL? [None] [Enter]
Display the list of known ftp servers? Y
```

We choose the next FTP server.

```
Server? ("IP address hostname, list #, 'done' or '?') 27
Using ftp.bytetime.net / pub / OpenBSD Oldenburg, Germany
Does the server support passive mode ftp? [Yes] [Enter]
Server directory? [Pub/OpenBSD/4.1/i386] [Enter]
Login? [Anonymous] [Enter]
```

Sets are displayed for selection. In this example, choose them all.

```
Set name? (Or 'done') [bsd.rd] all
Set name? (Or 'done') [done] [Enter]
Ready to install sets? [Yes] [Enter]
```

If everything is installed, you done one.

```
Location of sets? (Cd disk ftp http or 'done') done
```

The SSH server must be started. The NTP server is not needed here.

```
Start sshd (8) by default? [Yes] [Enter] Start ntpd (8) by default? [No] [Enter]
```

A graphical user interface is activated.

```
Do you expect to run the X Windows System? [No] Y
Change the default console to com0? [No] [Enter]
```

The time zone must be indicated accordingly.

```
What time zone are you in? ('?' For list) MET
```

When the installation was successful and the system will prompt the system is shut down.

```
Host ~ # halt
```

The virtual machine is restarted.

```
Host ~ $ qemu openbsd-hd.img
```

as a system administrator (root) After logging in, the X-server started with startx.

```
Host ~ # startx
```

To install software, first the ports system is configured.

```
Host ~ # cd / tmp
Host ~ # ftp ftp://ftp.openbsd.org/pub/OpenBSD/4.3/ports.tar.gz
Host ~ # cd /
Host ~ # tar xzf / tmp / ports.tar.gz
```

It can be searched these ports tree, for example by the browser Firefox.

```
Guest ~ $ cd / usr / ports
Guest ~ $ make search key = firefox
```

After the change in the specified directory the installation will start.

```
Host ~ # cd usr/ports/www/firefox-i18n /
Host ~ # make install clean
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b-f openbsd-hd.img qcows openbsd-hd-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu openbsd-hd-01.ovl
```

NetBSD 5.0.2

Website: <http://www.netbsd.org>

Recommendation: Standard PC, 128 MB RAM, 5 GB hard drive.

NetBSD is a Unix, the BSD family and is distributed freely under the BSD license. NetBSD is used because of its good portability to almost any computer: On servers, workstations, desktop PCs, notebooks, PDAs and embedded systems.

Installation CD Image

For a lean server installation wealthy few GB of hard drive capacity.

```
Host ~ $ qemu-img create-f qcows netbsd.img 5G
Host ~ $ wget \
  ftp://iso.ee.netbsd.org/pub/NetBSD/iso/5.0.2/i386cd-5.0.2.iso
Host ~ $ qemu-hda-cdrom netbsd.img i386cd-5.0.2.iso-boot d \
  -Net nic, model = ne2k_isa-net user
```

After installing the restart occurs with these options.

```
Host ~ $ qemu-net netbsd.img nic, model = ne2k_isa-net user
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b netbsd.img qcow2 netbsd-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-net netbsd-01.ovl nic, model = ne2k_isa-net user
```

Mini-Appliance

Download: <http://wiki.qemu.org/Download> (Small NetBSD Image)

It can be used an image with a minimal installed system.

```
Host ~ $ wget http://nopid.free.fr/small.ffd.bz2
Host ~ $ bzip2-d small.ffd.bz2
Host ~ $ qemu small.ffd
```

Solaris 10 - 10/08 (Intel), OpenSolaris, Illumos

Download:

- <http://www.sun.com/software/solaris/get.jsp> (90 days available)
- <http://www.opensolaris.com/get/index.jsp>
- <http://www.illumos.org>

Recommendation: Standard PC, 512 MB RAM, 15 GB hard drive.

The company Sun Microsystems (now Oracle Corporation), has been in decades of experience with hardware and software server area. Developed by Sun Microsystems Solaris operating system based on BSD and is especially high availability systems to be found. Since January 2005 there is the OpenSolaris project, that source code of Solaris as open-source development. In this example, Solaris 10 installed. To install Solaris you need a virtual disk.

```
Host ~ $ qemu-img create-f qcow2 solaris10.img 15G
```

You start the virtual machine with the following options.

```
Host ~ $ qemu-hda-cdrom solaris10.img solaris10.iso-boot d \
        Std-vga-m 256-net nic, model = rt18139-net user
```

At 256 MB of RAM, the installation is done in text mode, running under QEMU / KVM stable. First, the language is selected, then the network is activated. Turning on DHCP is recommended. Next one comes to the installation steps, each with the [F2]. The steps below name is IPv6, Kerberos security and disable service. It can be queried and the time zone settings for the country or region. The date and time are set and it is password for the user root to the demand. The remote service may be disabled. We choose the manual exchange of DVDs and a manual restart. The software license agreement is to agree and it is the geographical region and the character set selected. The installation program will ask for additional products to install. These options and disable the following query after the installation is answered in the ninth. Press [F4] defines the layout of the partitioning. We choose automatic partitioning by car from the layout. The question of the hanging of software from a remote computer denies it. The software packages will be installed then. After the installation, the virtual machine is started with the following options.

```
Host ~ $ qemu solaris10.img-m 256-usb-soundhw all \
        -Net nic, model = rt18139-net user
```

During the log you will be asked for the desired desktop (CDE or Java). The Java Desktop is based on Gnome. Software packages are installed from multiple sources, such <http://www.freeware.com>, <http://www.opensolaris.org> and <http://www.blastwave.org>. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b solaris10.img qcow2 solaris10-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-solaris10 01.ovl-m 256-usb-soundhw all \
        -Net nic, model = rt18139-net user
```

Solaris comes with the Solaris Zones already own virtualization solution with the operating system level to which a virtual machine under QEMU is used in the case (see http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulatoren/_Solaris_Zones).

Nexenta 2.0

Website: <http://www.nexenta.org>

Recommendation: Standard PC, 512 MB RAM, 10 GB hard drive.

Nexenta is based on a Solaris kernel. The software packages are managed with the convenient Debian packaging system. As a basis for the system Nexenta uses Debian GNU / Linux and Ubuntu.

```
Host ~ $ wget http://www.nexenta.org/releases/nexenta-core-platform_2.0-b104_x86.iso.zip
Host ~ $ unzip Nexenta_Core_platform_2.0-b104_x86.iso.zip
Host ~ $ qemu-img create-f qcow2 nexenta.img 10G
Host ~ $ qemu-hda nexenta.img \
        Nexenta-cdrom-core-platform_2.0_b104_x86.iso-boot d-m-512
```

After installing the restart occurs with these options.

```
Host ~ $ qemu-m 512 nexenta.img
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b nexenta.img qcow2 Nexenta 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-m 512 Nexenta 01.ovl
```

Darwin 8.01 (Intel)

Download: <http://www.opensource.apple.com/static/iso/>

<http://developer.apple.com/Darwin/>

Darwin is based on BSD with a Mach kernel and is the basis for Mac OS X. Darwin is open source (Apple Public Source License). The hybrid kernel XNU tried to take advantage of a monolithic kernel, with the advantages of a microkernel to connect. Darwin has been ported to PPC, x86 and ARM processors. All Programs for Darwin to run on Mac OS X. For Mac OS X development programs are not necessarily run on Darwin. To install I create a virtual hard disk.

```
Host ~ $ qemu-img create-f qcow2 darwin.img 10G
```

Furthermore, download the image file and extract it.

```
Host ~ $ wget \
```

```
http://www.opensource.apple.com/static/iso/darwinx86-801.iso.gz
Host ~ $ gunzip-darwinx86 801.iso.gz
```

When you install Darwin is to disable kvm or kqemu. You start the virtual machine with the following options.

```
Host ~ $ qemu-hda-cdrom darwin.img darwinx86-801.iso-boot d
```

The installation is done in text mode. Note the English keyboard. First, the hard drive is partitioned and formatted.

```
The following devices are available for installation: 1 QEMU HARDDISK @ disk0 (10.0GB) Enter 'shell' to drop into a shell Which device would you like
```

After the reboot, the created partition is formatted and installed the system.

```
The following devices are available for installation:
```

```
1. QEMU HARDDISK @ disk0 (10.0GB)
```

```
Enter 'shell' to drop into a shell
```

```
Which device would you like to install onto Darwin? 1
```

```
For partitioning the disk, you have the following choices:
```

```
1) Auto-partition the disk (Destroy alls disk contents)
```

```
2) Manually partition the disk using fdisk
```

```
3) Use existing partitions
```

```
Choice: 3
```

```
The following partitions are available:
```

```
/ Dev/disk0s1
```

```
Which will be the root partition?
```

```
 / Dev/disk0s1
```

```
Using
```

```
HFS) HFS + (journaled) file system
```

```
ufs) UFS filesystem
```

```
File system type: hfs
```

```
Would you like to do a clean install? (Yes / no) yes
```

```
Desired Volumename: darwin
```

After copying the software packages, the user password for the specified root. Then the computer name is specified. Furthermore, a user can be created. Then a restart with the following options is required.

```
Host ~ $ qemu-rtc darwin.img base = localtime \
-Net nic, vlan = 0, model = rtl18139-net user
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b darwin.img qcow2 darwin-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-darwin-rtc 01.ovl base = localtime \
-Net nic, vlan = 0, model = rtl18139-net user
```

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_x86-Architektur/_BSD-Unix-Systeme "

This page has been accessed 8547 times. This page was last updated on 3 November 2010 at 07:01 clock changed. Content is available under GNU Free Documentation License 1.2 .

Linux, Ubuntu 10.10 Maverick Meerkat, Debian, Fedora 14, Fedora, Red Hat Linux, OpenSUSE, download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / x86 / Linux]])

<<< | # # # | >>> | English

Contents	
1 Linux	
1.1 Ubuntu Desktop 10:10	
1.2 Ubuntu 10.04 LTS Server (32 -, 64-bit)	
1.2.1 Installation with QEMU / KVM	
1.2.2 Installation via PXE and QEMU / KVM	
1.2.3 Installation with the vmbuilder	
1.3 Debian GNU / Linux 5.04 Lenny (32-bit)	
1.3.1 Installation with QEMU	
1.3.2 Installation with Debian Live Linux	
1.3.3 Live-Helper	
4.1 Fedora 14 (32 -, 64-bit)	
SUSE 1.5	
1.5.1 Installing OpenSUSE with QEMU	
1.5.2 SUSE Studio	
1.6 Other Linux distributions	

Linux

Linux is a free multi-platform and multi-user operating system for computers, which is similar to Unix. The modular operating system developed by software developers around the world. Linux is really just the kernel. Free and commercial distributions that build their kernel configuration tools and put together software packages. It is impossible here to discuss all the distributions. A "hit parade" of Linux distributions is located at the URL <http://distrowatch.com/index.php?language=DE>. In the Linux guest should the ext3 file be used. The XFS file system is not recommended against it because it runs an intense write-caching. In case of sudden failure of the virtual machine, this can cause problems, especially when using *Virtio block devices*. Is set in the host system, the SVGA mode under X11, the X11 or Cirrus Vesa driver is configured. For optimum performance, a 16-bit color depth can be configured in the host and guest. When using a 2.6 kernel is the kernel option *clock = pit* to recommend. The 2.6 kernel is testing very strict real-time clock and QEMU does not emulate them exactly. The kernel option *clock = pit* can, for example, in /boot/grub/menu.lst be specified.

Ubuntu Desktop 10:10

Download: <http://www.ubuntu.com/getubuntu/download>

Recommendation: Standard PC, 512 MB RAM, Hard Drive: 10 GB

Ubuntu is a modern operating system based on the free Linux distribution Debian GNU / Linux. It's free and stable. Because of its easy operation and configurability, the support of developers and the community and its design has developed quickly into a popular Linux distribution and Linux beginners recommended. Many advanced Linux users use Ubuntu because of its adaptability. Ubuntu Linux by Canonical Ltd.. (South Africa) sponsored. Ubuntu uses GNOME as desktop environment, which was added later variants Kubuntu and Xubuntu rely on KDE or Xfce. The different variants differ only in the initial selection of the installed software packages and can therefore technically different configurations of a distribution are considered. For the virtual machine, the size of the hard disk is 10 GB.

```
Host ~ $ qemu-img create-f qcow2 d.img ubuntu-10g
```

You start the virtual machine with the following options.

```
Host ~ $ qemu-hda ubuntu-d.img-cdrom ubuntu-desktop-i386.iso-10.10 \
-Boot d-m 512-usb-sb16-USBDevice tablet soundhw
```

The desktop version of Ubuntu will start as a live CD. When the desktop appeared, the language is selected and the installation of the *Ubuntu install* started by clicking buttons. The graphical installer, the usual questions: What time zone should be set? What should be the keyboard layout? When partitioning, the system default will be taken. A special feature of Ubuntu is to establish a user for administrative tasks. In a personal computer can be your own name mentioned. After the security check begins to install the base system. When prompted to reboot to complete the instance and applies the following options.

```
Host ~ $ qemu-ubuntu d.img-usb-m 512-sb16-USBDevice tablet soundhw
```

To install new software from the Internet, one must first select the appropriate software sources. This is done in the menu *system, administration, software sources*. There you should activate all sources. One searches in the extensive software fundus (menu *system, configuration, Synaptic Package Manager*). If you have selected the appropriate packages, click on the *Apply* button. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b-f ubuntu-d.img qcow2 ubuntu-d-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu ubuntu-d-01.ovl-usb-m 512-sb16-USBDevice tablet soundhw
```

Ubuntu 10.04 LTS Server (32 -, 64-bit)

Download: <http://www.ubuntu.com/server/get-ubuntu/download>

Recommendation: Standard PC, 128 MB RAM, Hard Drive: 4 GB

Ubuntu is also a version for server use. The addition LTS stands for Long-Term Support. This means that is granted for five years, a support with appropriate updates. A server does not need a graphical desktop. Also for security reasons and because of resource consumption, this is not recommended. Therefore, the installation is in text mode.

Installation with QEMU / KVM

It is a virtual hard drive to create.

```
Host ~ $ qemu-img create-f qcow2 ubuntu-server.img 4G
```

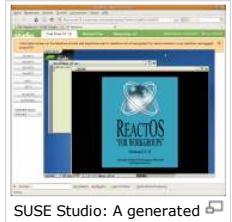
For the installation reaches the default value of 128 MB of RAM. For the 32-bit version of QEMU or KVM starts with the following options.

```
Host ~ $ qemu-hda ubuntu-server.img \
-Cdrom ubuntu-10.04-server-i386.iso-boot d
```

For the 64-bit version of the following options are applicable.

```
Host ~ $ qemu-system-x86_64-cpu-enable-kvm qemu64 \
-Bda ubuntu-server.img-boot d \
-Cdrom ubuntu-10.04.1-server-amd64.iso
```

It starts the installation in text mode. Initially, the language, country and keyboard will be selected. After the hardware detection by the network DHCP server is configured. After



entering the computer name, the disk is partitioned. For simplicity, we assume the default setting. For production use, the partitioning is to plan more accurately, since depend on the security and reliability of the system. The system time must be set to UTC. A special feature of Ubuntu is to define a user with admin rights. After a restart takes place with the following options:

```
Host ~ $ qemu-ubuntu server.img
```

For the 64-bit version of the following options are applicable.

```
Host ~ $ qemu-system-x86_64-enable-kvm-hda ubuntu-server.img
```

It logs in with the specified user account. prefer a system administrator (*root*) Who wants to sign, gives the command *sudo passwd root* user password *root*.

```
Guest ~ $ sudo passwd root
Password: *****
Enter new UNIX password: *****
Retype new UNIX password: *****
```

If you want to change network settings, it fits file */etc/network/interfaces* to the. After that, the network setting with */init.d/networking restart* to restart */etc*. Who wants to install the console software packages, the configuration file */etc/apt/sources.list*. These are lines to comment out all the *cdrom*. For all other lines with *deb* at the beginning, however, is the # comment characters removed. After which the package list to be read again.

```
Guest ~ $ sudo apt-get update
```

The software packages are updated.

```
Guest ~ $ sudo apt-get dist-upgrade
```

The kernel has been updated, a reboot of the system with *reboot* required. Subsequently, packets, such as *ssh* is *installed*.

```
Guest ~ $ sudo apt-get install ssh
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b ubuntu-server.img \
        -F qcow2 ubuntu-server-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu ubuntu-server-01.ovl
```

For the 64-bit version of the following options are applicable.

```
Host ~ $ qemu-system-x86_64-cpu=qemu64 kvm-hda ubuntu-server-01.ovl
```

If you want a nested virtualization, the *option-cpu qemu64*, + *svm* necessary. The host system must adjust to AMD processors.

```
Host ~ $ qemu-system-x86_64-cpu qemu64, + svm-smp 2 \
        -Hda ubuntu-server-01.ovl
```

Installation via PXE and QEMU / KVM

The installation of Ubuntu can be made by PXE (see http://qemu-buch.de/d/Netzwerkoptionen/_Netzwerkdienste). First create a directory to download the netboot archive and uncompress.

```
Host ~ $ sudo mkdir netboot /
Host ~ $ cd ~ / netboot
Host ~ $ sudo wget \
http://archive.ubuntu.com/ubuntu/dists/lucid/main/installer-i386/current/images/netboot/netboot.tar.gz
Host ~ $ sudo tar xzvf netboot.tar.gz
```

In any directory is generated for the virtual disk to install QEMU or KVM and then run with the appropriate options.

```
Host ~ $ qemu-img create-f qcow2 ubuntu.img 10G
Host ~ $ qemu-hda-boot ubuntu.img n \
-Net nic-net user = tftp / netboot / boot file = pxelinux.0
```

Installation with the vmbuilder

If the host system is Ubuntu Linux, script is the creation of a virtual machine with the Python *vmbuilder* possible. The installation is done under Ubuntu using a command line.

```
Host ~ $ sudo apt-get install python-vm-builder
```

In the following example, a virtual machine with Ubuntu is generated 10:04 Lucid Lynx. In the current directory to the script specifies the subdirectory *ubuntu-kvm* with the necessary files.

```
Host ~ $ sudo vmbuilder kvm ubuntu - suite intrepid
```

To start the virtual machine, change to the created subdirectory and calls the script *run.sh*.

```
Host ~ $ cd ubuntu-kvm
Host ~ $ ./ Run.sh
```

The Python script is *vmbuilder libvirt* in section above (see http://qemu-buch.de/d/Managementtools/_libvirt-Tools/_Die_Managementtools).

Debian GNU / Linux 5.04 Lenny (32-bit)

Download: <http://www.debian.org/distrib/>

Recommendation: Standard PC, 128 MB RAM, Hard Drive: 2 GB (Server)

Debian GNU / Linux is a free Linux distribution that is compiled exclusively from free software. Debian is a reliable server operating system and serves as a stable foundation for many other Linux distributions such as Ubuntu. Debian is available for eleven processor architectures.

Installation of QEMU

A virtual hard disk size of 2 GB is sufficient.

```
Host ~ $ qemu-img create-f qcow2 debian.img 2G
```

For the installation reaches the default value of 128 MB of RAM.

```
Host ~ $ qemu-hda debian.img \
        -Cdrom debian-504-i386-netinst.iso-boot d
```

At the beginning of the installation language, country and keyboard will be selected. After the hardware detection by the network DHCP server is configured. Then the computer name is entered and it will partition the hard drive. For simplicity, we assume the default setting. The system time is set to UTC. It restarts with the following options:

```
Host ~ $ qemu debian.img
```

If you want to change network settings, it fits file `/etc/network/interfaces` to the. It is the network setting `/init.d/networking restart` to restart `/etc`. Who wants to install the console software packages, the configuration file `/etc/apt/sources.list`. These are lines to comment out all the `cdrom`. For all other lines with `deb` at the beginning, however, is the # comment characters removed. After which the package list to be read again.

```
Host ~ # apt-get update
```

An update of the software packages is recommended.

```
Host ~ # apt-get dist-upgrade
```

The kernel has been updated in this case, a reboot of the system with `reboot` required. Subsequently, packets, such as `ssh` is installed.

```
Host ~ # apt-get install ssh
```

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b debian.img qcow2 debian-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-debian 01.ovl
```

Installation with Debian Live Linux

With Debian-Live makes it possible to generate a live system based on Debian. Database systems allow it to boot operating systems from CDs, DVDs or USB sticks. This allows operating systems to test without changing the disks. Other applications include forensic analysis, the repair of installed systems and the transfer of a physical operating system installed on a virtual machine (Physical to virtual). Unlike other live systems Debian Live is not limited to the i386 architecture. Debian Live further enables the adaptation of the live system to your needs. It can, for example, the Debian *Stable* branches, *testing* and *unstable* to choose and install different software packages for CDs, DVDs or USB sticks to create images. With Debian Live, it is also for QEMU or KVM virtual machines generate. The installation of the necessary packages is `live-helper` and `cdebootstrap` Ubuntu using a command line does under.

```
Host ~ $ apt-get install live-helper cdebootstrap
```

The package includes `live-helper` shell scripts whose names begin with `lh_` and in the `/usr/bin/` are stored. The main scripts are `lh_config`, `lh_build` and `lh_clean`. The script sets the current directory `lh_config` the subdirectory `config`. The files in this directory can be adapted to configure the live system. The script generates `lh_build` the image corresponding to the configuration. This is the first step (bootstrapping) downloaded the basic system and stored in a directory. It is created based on that directory, a chroot environment. It will install the required additional packages and configure. At the end of the installation directory is compressed, install a boot loader and create the image. This step is called `binary`. With the script `lh_clean` cleaned up the results of a build. Since the live system using a chroot environment is generated, can only live systems compatible to the host system architecture can be formed. The generation of live systems for other architectures (SPARC, MIPS, PowerPC, ...) can, in QEMU (`qemu-system-*`) the virtual machines in place. To build the creation of a subdirectory is recommended.

```
Host ~ $ mkdir ~ / my-debian-live
Host ~ $ cd ~ / my-debian-live
```

The following command generates an ISO image with the default settings.

```
Host ~ $ lh_config-a i386 && sudo lh_build
```

With QEMU or KVM that image is tested.

```
Host ~ $ qemu-cdrom binary.iso
```

The image `binary.iso` can be burned CD Live as, for example with the following command.

```
Host ~ $ wodim binary.iso
```

Another is to build a place, it must be cleaned up `sudo lh_clean` with first.

```
Host ~ $ sudo lh_clean
```

The script uses `chroot lh_build`. For the chroot environment, some subdirectory mount points created in the `chroot`, the build process to be deleted after the. If the build process is killed, they must mount point with the `umount` command will be appended. The configuration of the chroot environment is in the `subdirectory Stage` saved. If you want to complete the build process, the empty directory, this directory is also deleted. For generating images for USB flash drives or hard drives is `lh_config` with the `-b usb hdd apply`.

```
Host ~ $ lh_config-a i386-b usb-hdd && sudo lh_build
```

The generated image can be used as QEMU or KVM virtual disk. Since the applied file system is `squashfs` compressed file system, execution speed is low. The use of the generated image in QEMU or KVM is better suited to test the live system.

```
Host ~ $ qemu-hda binary.img
```

Other options can, for example, the work environment `xfce` the testing branch Lenny install, and configure the German keyboard layout.

```
Host ~ $ lh_config-a i386-b usb-hdd-d lenny xfce-p \
  - Boot append keyb = de
Host ~ $ sudo lh_build
```

The configuration of the live system on the options of `lh_config` is cumbersome and inflexible. Better the fit of the configuration files in the directory `config` between the calls to `lh_config` and `lh_build`. Most of the options `lh_config` have a direct equivalent in one of the configuration files. For example, stores the file `config / binary` in the variable `LH_BOOTAPPEND` boot options. To further configuration in the file `config / binary`, the host name and user-defined name of the.

```
# Config / binary - options for live-helper (7), binary stage ... LH_BINARY_IMAGES = "usb-hdd" LH_BOOTAPPEND_INSTALL = "keyb = de)" LH_HOSTNAME = "No:
```

The file `config / bootstrap` stored in the variable `LH_ARCHITECTURE` the architecture and in the variable `LH_DISTRIBUTION` distribution. Furthermore, here, the sections of the distribution are defined.

```
# Config / bootstrap - options for live-helper (7), bootstrap stage
...
LH_ARCHITECTURE = "i386"
LH_DISTRIBUTION = "lenny"
LH_SECTIONS = "main contrib non-free"
...
```

The file `config / chroot` stored in the variable `LH_PACKAGES_LISTS` the list of installed software packages.

```
# Config / chroot - options for live-helper (7), chroot stage
...
LH_PACKAGES_LISTS = "xfce"
LH_PACKAGES = "openssh-client openssh-server"
```

The required package lists a new file in `config / chroot_local-package lists` are also listed.

```
# Config / chroot_local-package lists my_Packages /
mc iceweasel qemu
```

If the configuration is complete, a new build process is started.

```
Host ~ $ sudo lh_clean & & sudo lh_build
```

Would like this configuration for the generation of other live systems use one is *config* directory to secure it.

```
Host ~ $ tar czvf config.tar.gz config
```

Opportunities are also created by adapting the chroot environment before its image is generated. This is possible because the script *lh_build* a wrapper script, the script *lh_bootstrap*, *lh_chroot* *lh_binary* and successively calls the. Will take the script the following command line entered *lh_build*, directory is initially created in the chroot environment *chroot*.

```
Host ~ $ sudo lh_bootstrap & & sudo lh_chroot
```

The directory contains the *chroot* directory tree of the future live system.

```
Host ~ $ ls chroot
```

```
bin boot dev etc home lib initrd.img initrd.img.old
media mnt opt proc root sbin selinux srv sys tmp
usr var vmlinuz vmlinuz.old
```

It is possible to create files in this chroot environment copy, so they are used in the live system. With the command *chroot* log into this chroot environment, and it can install additional packages.

```
Host ~ $ sudo chroot chroot
```

```
Host ~ # apt-get install netcat mc lynx
```

Furthermore, let the configuration files in / *etc* adapt. With *exit*, leave the chroot environment.

```
Host ~ # exit
```

```
Host ~ $
```

Are all adjustments completed, the image is generated.

```
Host ~ $ sudo lh_binary
```

Live-Helper

Website: <http://live-build.debian.net/cgi-bin/live-build>

Live Helper is a web application for the Debian Live Project. They typically generate binary images for CD / DVDs and hard drives.

Fedora 14 (32 -, 64-bit)

Website: <http://fedoraproject.org>

Recommendation: Standard PC, 512 MB RAM, Hard Drive: 10 GB

The Linux distribution Fedora is Red Hat Linux from the former arose and enjoys great popularity. Fedora is related to the Red Hat Enterprise Linux (<http://www.redhat.de/rhel/>) and its derivative CentOS (<http://www.centos.org>). For the virtual machine, the size of the hard disk is 10 GB.

```
Host ~ $ qemu-img create-f qcows fedora.img 10G
```

For the 32-bit version of the command is applied.

```
Host ~ $ qemu-hda-cdrom Fedora fedora.img-14-i686-Live-Desktop.iso \
-Boot d-m 512-net user-net nic, model = PCnet
```

The 64-bit version you start with *qemu-system-x86_64*. Requires the 64-bit live DVD.

```
Host ~ $ qemu-system-x86_64-hda fedora.img-cdrom Fedora-14-x86_64-DVD.iso \
-Boot d-m 512-net user-net nic, model = PCnet
```

The desktop version starts the same as a live CD. Keyboard selection is on the language and the desktop appeared, the installation is clicking on the icon *Install to Hard Drive* launched by. The installation program provides the usual questions after the speech, after the host name and time zone. When partitioning, the default is assumed. When prompted to reboot to complete the system and launches the virtual machine with the following options:

```
Host ~ $ qemu fedora.img-m 512-net user-net nic, model = PCnet
```

The 64-bit version you start again with *qemu-system-x86_64*.

```
Host ~ $ qemu-system-x86_64 fedora.img-m 512-net user-net nic, model = PCnet
```

After the first boot from the hard drive is a user to create and set the system time. For newly installed Fedora system to update the software packages is recommended.

```
Host ~ # yum check-update
Host ~ # yum update
```

Yum install software packages and information on the package name installed.

```
Host ~ # yum install package
```

If the package name is unknown, we look for packages with *yum search*.

```
Host ~ # yum search package_name
```

A software package *yum remove* removed.

```
Host ~ # yum remove package_name
```

The following command lists all installed packages.

```
Host ~ # rpm-qa
```

The management of software packages can also be done with a GUI. This is to install *yumex*.

```
Host ~ # yum install yumex
```

It is started *yumex*.

```
Host ~ # yumex
```

A largest package selection allows the integration of RPMFusion repositories. Instructions can be found at <http://rpmfusion.org/Configuration>.

It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b fedora.img qcows fedora-01.ovl
```

The start is the 32-bit version of the overlay file with the following options:

```
Host ~ $ qemu fedora-01.ovl-m 512-net user-net nic, model = PCnet
```

For the 64-bit version is used `qemu-system-x86_64`.

```
Host ~ $ qemu-system-x86_64 fedora-01.ovl-m 512-net user-net nic, model = PCnet
```

SUSE

Installation of OpenSUSE with QEMU

Download: <http://de.opensuse.org/Download>

Recommendation: Standard PC, 512 MB RAM, Hard Drive: 10 GB

The roots of OpenSUSE go back to the beginning of the nineties, when Linux will be loaded on about 50 floppy disks from the Internet but had a few users at all had internet access. The former SuSE GmbH (society for software and systems development), Linux disks into one package. SuSE is now a Linux distribution by Novell or its subsidiaries Suse Linux GmbH (Nuremberg). With the OpenSUSE project, the development of SuSE Linux was made public. To install a virtual disk is created.

```
Host ~ $ qemu-img create-f qcows opensuse.img 10G
```

You start QEMU, or the Kernel-based Virtual Machine with the virtual hard drive and the installation DVD.

```
Host ~ $ qemu-hda-cdrom opensuse.img openSUSE-11.3-DVD-i586.iso \
-Boot d-m 512-usb-soundhw all
```

After the start screen of the OpenSUSE installation are the queries to the language, to accept the license agreement, if there is a new installation and the time zone. This OpenSUSE in the emulation runs faster, a minimum graphical system or the text mode is preferable. The proposal to partition and to the software selection can be assumed. After installing the software, the system is started from the hard disk.

```
Host ~ $ qemu opensuse.img-m 512-usb-soundhw all
```

It is the password for the user to enter `root`. Furthermore, the computer and the domain name should be specified. The default network settings (DHCP) are unchanged. It is recommended to enable the additional installation sources. Then, the authentication method (`/etc/passwd`, LDAP, NIS, Windows Domain) input. Here is the authentication `etc/passwd` to use `/`, which the local host application system is on. After this selection, a user can be created. Thereafter, the amendment of the hardware configuration is possible. Other configurations are using the graphical tool `YAST`. To install software packages, only the on-line sources should be indicated. This is done in `YAST Installation Source` under. Then you look with `Yast software install` under or `delete` the desired software packages and installs them. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b opensuse.img qcows0 fedora-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-01.ovl opensuse-m 512-usb-soundhw all
```

SUSE Studio

Website: <http://susestudio.com>

With SUSE Studio, imagine your own appliance together in the Web browser. To be able to use SUSE Studio, you need an account. This is obtained via an invitation. After configuring the appliance to Load a RAW image for QEMU / KVM or USB stick, as a VMware image, as an ISO image (Live CD) or downloaded as a Xen guest. You can download and test the appliance without the browser.

Other Linux distributions

- http://qemu-buch.de/d/Gast-Systeme/_x86-Architektur/_Linux/_Weitere

<<< | # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_x86-Architektur/_Linux "

This page has been accessed 17,848 times. This page was last updated on 28 January 2011 at 07:22 clock changed. Content is available under GNU Free Documentation License 1.2 .

Unix-like operating systems, Minix3, Plan 9, download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / x86 / Unix-like operating systems]])

<<< | # # # | >>> | English

Unix-like systems

Minix3

Website: http://www.minix3.org

Recommendation: standard PC (default), 128 MB RAM, hard disk: 1 GB.

Minix is a free Unix-like operating system. It was developed by Andrew S. Tanenbaum at the Free University of Amsterdam as a teaching tool.

Minix was a motivation for, among other things, that the source code of Unix was not for teaching purposes at universities. A final system is available for download.

```
Host ~ $ wget \ http://www.minix3.org/download/minix3_1_1_small_vmware_256MB_1GB.zip host ~ $ unzip minix3_1_1_small_vmware_256MB_1GB.zip host ~ $ qemu
```

After starting you logged in as root without a password. The German keyboard is *loadkeys /usr/lib/keymaps/german.map* set up under. End the system with *shutdown -h now*. It is recommended that you install and configure the virtual machine to shut down and one or more overlay to create files to image to make changes to protect the base.

```
Host ~ $ qemu-img create-b minix311_256MB_1GB/MS-DOS.vmdk \
-F qcow2 minix311_256MB_1GB/MS-DOS-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu minix311_256MB_1GB/MS-DOS-01.ovl
```

Plan 9

Download: http://plan9.bell-labs.com/plan9/

Recommendation: Standard PC (default), from 32 MB of RAM, hard disk: 1 GB.

Plan 9 is since 1985 in the AT & T Bell Laboratories. It is an advanced multi-user operating system which is being developed as open source and not on Unix source code builds. The concept is to split it efficiently to the computers on the network. It processes can be mirrored to other computers, to ensure availability even when individual components fail. Plan 9 is particularly suitable for use in distributed network system on different hardware platforms. It is not meant for the end user. It has a few missing drivers and commercial applications.

```
Host ~ $ qemu-img create-f qcow2 plan9.img 1G
```

To download the ISO image of the installation CD is available. You will download this image and unpacked it. Here are the Linux commands.

```
Host ~ $ wget http://plan9.bell-labs.com/plan9/download/plan9.iso.bz2
Host ~ $ bunzip2 plan9.iso.bz2
```

QEMU is or Kernel-based Virtual Machine started with these options.

```
Host ~ $ qemu-hda-cdrom plan9.img plan9.iso boot d-m-32
```

The CD is both live and installation CD. To install the selected *first*

1. Install Plan 9 from this CD
2. Plan 9 boot from this CD

On most issues with [Enter] accepted the default.

```
use DMA for IDE drives [yes]: [Enter]
mouse port is [ps2]: [Enter]
```

The resolution is increased to 1024x768.

```
vgasize [640x480x8]: 1024x768x8
monitor is [xga]: [Enter]
```

It starts the graphical user interface *Rio* with the installer.

```
Task to do [configfs]: [Enter]
File system [fossil]: [Enter]
Task to do [part disk]: [Enter]
```

It selects the virtual hard disk.

```
Disk to partition: sdC0
Install mbr (y, n) [no default]: y
```

Is then used to partition the program started *fdisk*. Since the entire hard disk is to be used, *w* and *q* enter.

```
>>> w
>>> q
Task to do [prepdisk]: [Enter]
Plan 9 partition to subdivide [/ dev/sdC0/plan9]: [Enter]
```

Even with the division of the partition and enter *q w*.

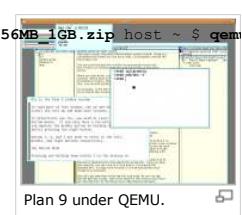
```
>>> w
>>> q
Task to do [fmtfossil]: [Enter]
Fossil partition to format [/ dev/sdC0/fossil]: [Enter]
Task to do [mountfs]: [Enter]
Fossil partition [/ dev/sdC0/fossil]: [Enter]
Task to do [configdist]: [Enter]
Distribution is from [local, net [local]]: [Enter]
Task to do [mountdist]: [Enter]
```

When asked about the source of the software installation CD must be reported.

```
Distribution disk [no default]: / dev/sdD0/data
Location of archives [browse]: [Enter]
```

It can be checked using shell commands, the software sources. Terminated with *exit* to the shell again.

```
% Exit
```



It will install the system on the hard disk.

Task to do [copydist]: **[enter]** Task to do [boot setup]: **[Enter]** to enable boot method (floppy, plan9, win9x, winnt): **plan9** Install the 9 master boot p:

The installation is completed and the system is restarted.

```
Host ~ $ qemu-hda plan9.img-m 32
```

Plan 9 allows booting over the network and on the local hard disk.

```
local) [local! S/sdC0/fossil #] root is from (tcp: [Enter]
```

As a standard user is entering *glenda*.

```
user [none]: glenda
```

To operate the graphical user interface *Rio* is a three-button mouse is necessary. In a two-button mouse middle button is emulated by pressing both mouse buttons. In order to open a shell window, with the right mouse button on the desktop and click on the shortcut menu select *New*. The cursor becomes a cross and it moves with the right mouse button on the desktop is a rectangle. A window will appear with the rc shell. To configure DHCP to leave via the network, the command is *ip / ipconfig apply*. Name resolution via DNS is *ndb / r enter dns*.

```
Host $ ip / ipconfig  
Guest $ ndb / dns-r
```

Plan 9 is *fshalt* down with.

```
Guest $ fshalt
```

An Introduction to Plan 9 is at the URL <http://www.lulu.com/content/406379> available. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b plan9.img qcow2 plan9-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-hda-plan9 01.ovl-m 32
```

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_x86-Architektur/_Unix-% C3% A4hnliche_Betriebssysteme "

This page has been accessed 4123 times. This page was last updated on 3 April 2010 at 11:32 clock changed. Content is available under GNU Free Documentation License 1.2 .

OpenVMS-like operating systems, VMS (Virtual Memory System), 0.4 FreeVMS O3ON, dcl, download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / x86 / OpenVMS-like operating systems]])

<<< | # # # | >>> | English

OpenVMS-like systems

VMS (Virtual Memory System) is an operating system the computer manufacturer Digital Equipment Corporation (DEC). It was at the time of publication (1978) an extremely advanced 32-bit operating system, which was multi-user and multi-tasking. The company DEC was later acquired by Compaq, which in turn was acquired by Hewlett Packard. Meanwhile, VMS was renamed OpenVMS. run on OpenVMS on mobiles and SMS solutions in banking, stock market systems operate. Using the URL http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulatoren/_PersonalAlpha Windows, the installation of OpenVMS on the emulator alpha for personnel described.

FreeVMS 0.4

Website: <http://www.freevms.net>

Recommendation: 64-bit PC, 64 MB RAM, hard disk: 1 GB.

FreeVMS is an OpenVMS-like operating system that is freely available under GPL. It consists of a POSIX kernel and a DCL command line interpreter. The only currently supported architecture is x86_64. On the website of disk images are available for download. After downloading the compressed images unpack them. Here are the Linux commands.

```
Host ~ $ wget http://www.freevms.net/IMG/bz2/freevms-img.bz2
Host ~ $ bzip2-d-freevms img.bz2
```

QEMU is or Kernel-based Virtual Machine started with the image file.

```
Host ~ $ qemu-system-x86_64 freevms-img-rtc base = localtime
```

It is recommended to shut down after installing and configuring the virtual machine and create overlay files to protect the base images to make changes.

```
Host ~ $ qemu-img create-f-b freevms-img-qcow2 freevms 01.ovl
```

Is started by the overlay files with the following options:

```
Host ~ $ qemu-system-x86_64-img-freevms 01.ovl-rtc base = localtime
```

If the text console of the instance does not appear in a window, *curses* is the option to apply.

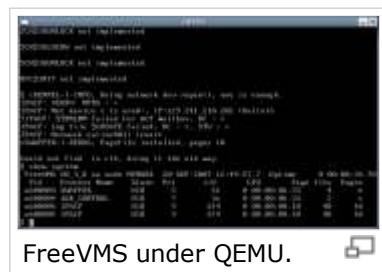
```
Host ~ $ qemu-system-x86_64-img-freevms 01.ovl-rtc base = localtime-curses
```

O3ON

Website: <http://www.o3one.org>

Recommendation: Standard PC (default), 32 MB RAM, hard disk: 40 MB.

O3ON is an OpenVMS-like operating system with UNIX features. Everything is in the kernel object



FreeVMS under QEMU.



O3ON under QEMU.

(threads, processes, devices, files, ...). o3zone (also called Ozone) is under the GPL. An image with pre-installed system is available.

```
Host ~ $ wget http://www.o3zone.org/downloads/hd40meg.img.gz
Host ~ $ gunzip hd40meg.img.gz
Host ~ $ qemu hd40meg.img
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_x86-Architektur/_OpenVMS-% C3% A4hnliche_Betriebssysteme "

This page has been accessed 4373 times. This page was last updated on 23 November 2010 at 11:52
clock changed. Content is available under GNU Free Documentation License 1.2 .

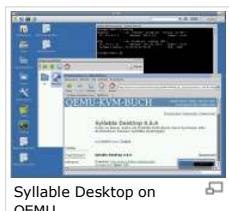
SkyOS, Syllable Desktop 0.6.6, Haiku, BeOS, Zeta, MenuetOS, AROS, Amiga, download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / x86 / Exotic]])

<<< | # # # | >>> | English



Haiku is open source and compatible with BeOS.



Syllable Desktop on QEMU.



SkyOS under QEMU.



The one-floppy OS MenuetOS.



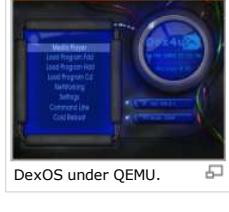
AROS under QEMU.



BOS under QEMU.



Cosmos under QEMU.



DexOS under QEMU.

Contents

An exotic

- 01.01 Haiku
- 2.1 Syllable Desktop
- SkyOS 1.3
- MenuetOS 4.1 (64-bit)
- 1.5 AROS
- 06/01 BOS
- 1.7 Cosmos
- 1.8 DexOS
- 1.9 JNode
- 1:10 loosely thos
- 1:11 MikeOS
- 1:12 MonaOS
- 1:13 SharpOS
- 1:14 Visopsys
- 1:15 Whitix

Exotic

Haiku

Download: <http://haiku-os.org/downloads>

Recommendation: Standard PC (default), 128 MB RAM, hard disk: 1 GB.

Haiku is an open-source project with the aim that nachzuprogrammieren BeOS operating system and expand. Haiku is still in alpha phase. To test the sleek and user-friendly Haiku, invites you to the download section Download the compressed ISO image and unpacked it. Here are the Linux commands.

```
Host ~ $ wget http://get-haiku.crisu.de/releases/r1alpha2/haiku-r1alpha2-iso.zip host ~ $ unzip-haiku-r1alpha2 iso.zip
```

A virtual hard disk with two GB is sufficient.

```
Host ~ $ qemu-img create-f qcows haiku.img 2G
```

You start QEMU / KVM with the following options.

```
Host ~ $ qemu-boot d-hda haiku.img \
Cdrom-haiku-r1alpha2.iso
```

After the voice, and keyboard to start with a click on the button *Run Installer* to install. After reading the instructions you go to the next step to continue in the *Drive Setup* window, choose the virtual hard disk (*/dev/disk/ata/0/master/raw*) and select the menu item *partition, initialize, Be File System*. After initializing closes the window and *Drive Setup* are virtual hard disk as the medium to the target. *Start* with one begins with the installation. After successful installation, you boot the virtual machine.

```
Host ~ $ qemu-usb-haiku.img soundhw all
```

It is recommended to shut down after configuring the virtual machine and one or create several overlay files.

```
Host ~ $ qemu-img create-f-b haiku.img qcows haiku-01.ovl
```

QEMU or KVM will start with the following options:

```
Host ~ $ qemu-usb-haiku-01.ovl soundhw all
```

Software for haiku can be found at the URLs <http://www.haikuware.com> and <http://tiltos.com/drupal/>. An installation guide can be found at the URL <http://www.haiku-os.org/get-haiku/installation-guide>.

Syllable Desktop

Download: <http://web.syllable.org/pages/get-Syllable.html> (Basic CD)

Recommendation: Standard PC (default), from 32 MB of RAM, hard disk: 1GB

Syllable Desktop is a fast desktop operating system which requires very little resources. It works well for virtual machines. Syllable Desktop is based neither on Unix, Linux or another operating system. The development of the kernel is based on the POSIX standard. Syllable Desktop is released under the GPL. There continues to Syllable Server. This variant is based on a Linux kernel and is not binary-compatible with the Syllable desktop. For Syllable Desktop is a virtual hard drive from a GB is sufficient.

```
Host ~ $ qemu-img create-f qcows Syllable.img 1G
```

After downloading the image of the installation CD and unpacked, you start QEMU / KVM with the following options.

```
Host ~ $ qemu-boot d-hda Syllable.img-no-kqemu \
-Cdrom-SyllableDesktop 0.6.6.i586.iso
```

The installation program is important to note that an English keyboard is set. The installation begins with the [i]. This is followed by partitioning the hard disk.

```
Press 'i' or 'I' to install now: i
Do you need to create a new partition? (Y / n) y
```

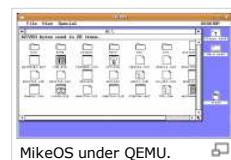
In the dialog box is the device */dev/disk/ata/hda/raw* activated and the button is clicked *partition*. In the next dialog box will be under */dev/disk/ata/hda/0* in the columns, the values *63, Syllable - AFS* and selected *2097151st* Then the disk is formatted. Since only one partition available for selection, the next question with [a] be answered.

```
Enter the full path of the letter or partition
that you would like to install Syllable onto,
or "ls" To see the partition List Again: a
Are you sure? Type "yes" to continue: yes
```

At the end of the installation record is the GRUB boot loader in the *master boot* set up.

```
The base packet has been installed. [Enter]
Auto Generating GRUB menu.lst ... done
Press "l" to view the menu.lst file, "e" to edit it manually,
"?" for an explanation of the menu.lst file,
or ENTER to continue with the installation: [Enter]
Press "m" to install GRUB automatically in the master boot record of
disk / dev / disk / ata / hda / raw,
Press "p" to install GRUB on partition automatically / dev/disk/ata/hda/0,
or press ENTER to skip this step: m
Please press "b" to reboot your computer: b
```

With these options, the system is started.



```
Host ~ $ qemu Syllable.img
```

It logs in as system administrator *root* with the password *root*. Preferences under language, the screen resolution, keyboard, and adapted. Since the network was configured by DHCP automatically, for example, the Web browser *Webster* used immediately. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b Syllable.img qcows2 Syllable 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-Syllable 01.ovl
```

SkyOS

Download: <http://www.skyos.org> (fee-Beta Version)

Recommendation: Standard PC (default), 512 MB RAM, hard disk: 2GB

At present, the continued development of SkyOS is unclear. There are discussions SkyOS to publish as open source. SkyOS is likely to get a Linux or NetBSD kernel.

SkyOS is a graphical operating system for modern x86 PCs that is designed for the desktop. SkyOS is not based on Microsoft Windows, Linux or Unix. It was written from scratch once again and is under a proprietary license. SkyOS has a POSIX-compatible programming interface. At the time of hardware support is limited. Followers of the operating system praise the high speed and the simplicity of the system. Who is on paid beta testing involved will receive the final version later for free. After logging in and pay you get a serial number and download link. Then unpack the RAR archive with the installation CD. Here are the Linux commands.

```
Host ~ $ rar e-B6947.rar SkyOS
```

There is created a virtual hard disk with a size of 2 GB.

```
Host ~ $ qemu-img create-f qcows2 SkyOS.img 2G
```

You start QEMU, or the Kernel-based Virtual Machine with the virtual hard drive and the installation CD.

```
Host ~ $ qemu-hda-cdrom SkyOS.img retail.iso-boot d \
M-512-rtc base = localtime \
Soundhw-es1370-vga std \
-Usb-net nic, model = rtl18139-net user
```

After entering his registration name and serial number, SkyOS boot as a live CD. You logged in as *admin* without the password. SkyOS to be installed on the hard drive, click on the *Install* icon on the *SkyOS*. Then you choose the language and keyboard layout and confirmed the license. The partitioning of the hard disk is done with the *disk manager*. For this you choose the option *Selected harddisks k / dev/hd0 As*. With the *Create Partition* button puts you to a new partition. The size shall be selected as the type and *SkyOS*. Then a restart is required. After logging into the live system, you click again on the icon *Install SkyOS* and through all the steps again to partition off to. We choose the partition and format it with the file system *skyf*. It allows software packages will be selected for installation. At the end of the installation, the boot loader is installed. For this you choose as a target *MBR* (Master Boot Record) from. After installation, the system is rebooted.

```
Host ~ $ qemu-m 512 SkyOS.img net nic, model = rtl18139-net user
Rtc-base = localtime-std-vga soundhw es1370-usb-no-kqemu
```

The options are in the Kernel-based Virtual Machine use.

```
Host ~ $ kvm-m 512-rtc SkyOS.img base = localtime-soundhw es1370 \
Std-vga-usb-net nic, model = rtl18139-net user-no-kvm
```

Further configuration of SkyOS is a simple way with the *system manager*. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-f-b SkyOS.img qcows2 SkyOS-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-SkyOS 01.ovl-m 512-localtime-rtc base = soundhw es1370 \ std-vga-usb-net nic, model = rtl18139-net user-no-kqemu
```

MenuetOS (64-bit)

Download: <http://www.menuetos.net/download.htm>

Recommendation: Standard PC (default), CPU: x86 64-bit, 512 MB RAM, no HDD

The operating system MenuetOS is written entirely in x86 assembly language. It fits on a floppy disk. MenuetOS is not based on UNIX or the POSIX standard. It is based not on a different operating system. The 32-bit version is hardly developed further. Therefore, the 64-bit version is recommended. To download a disk image will be available. You will download this image and unpacked it. Here are the Linux commands.

```
Host ~ $ wget http://www.menuetos.be/download.php?CurrentMenuetOS
Host ~ $ unzip *.zip *
```

It *is-system-x86_64 qemu* started with the following options.

```
Host ~ $ qemu-system-x86_64-fda M64-095Q.IMG-m 512-rtc base = localtime
```

In the default configuration changes are not saved. For permanent storage in the host system, the disk image integrated and edited the configuration file. Here the example of the Unix-/Linux-Befehle that must be executed under the root user.

```
Host ~ $ sudo mount-o loop-M64 094G.IMG / mnt
```

After mounting the file *config.mnt* with an editor as *vi* processed. If the space on the disk image does not, this file is to copy to edit in a different directory and write back after editing.

```
Host ~ $ sudo vi / mnt / config.mnt
```

To adjust the screen resolution, *screen_resolution* the value changed behind. A 3 at the end of the hexadecimal number giving a resolution of 1024x768 pixels.

```
screen_resolution = 0x0000003 # 1 = 640x480 2 = 800x600
# 3 = 1024x768 4 = 1280x1024
```

The network will be activated and configured as follows.

```
network_card_enable = 0x00000001 # network card enabled network_ip 0x0f02000a = # # 10.0.2.2 10.0.2.15 network_gateway = 0x0202000a network_subnet_mas:
```

If any changes occurred, is unmount the disk image.

```
Host ~ $ sudo umount / mnt
```

It *is-system-x86_64 qemu* started with these options. It is recommended to shut down after configuring the virtual machine and one or create several overlay files.

```
Host ~ $ qemu-img create-b-f qcows2 094G.IMG M64 M64-094G-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-system-x86_64-fda M64-094G.ovl-m 512-rtc base = localtime
```

AROS

Website: <http://www.aros.org>
Download: <http://aros.sourceforge.net/download.php> # winaros

AROS is a desktop operating system to be compatible with AmigaOS 3.1 at the API level. An image with an installed system is downloaded and decompressed with 7-Zip.

```
Host ~ $ wget http://amidevcpp.amiga-world.de/WinAros/WinArosHD.7z
Host ~ $ p7zip-d WinArosHD.7z
Host ~ $ qemu WinArosHD.img
```

Icaros desktop LIVE! Is a live CD and is from the URL <http://vmwaros.blogspot.com/2009/03/icaros-desktop-live-is-complete.html> downloaded. After unpacking the following options can be applied.

```
Host ~ $ kvm-m 192-net nic, model = PCnet-net user \
Rtc-base = localtime-cdrom icaros-v110-unknown-i386.iso
```

There are also Icaros Desktop (<http://ve.icarosdesktop.org>). This is a AROS-Appliance for VMware with the OWB Web browser and other software.

BOS

Website: <http://bos.asmhackers.net>

The aim of BOS is simplicity. BOS is similar to DOS.

```
Host ~ $ wget http://bos.asmhackers.net/downloads/bos_0.04_img.zip
Host ~ $ unzip bos_0.04_img.zip
Host ~ $ qemu-fda-rtc bos_0.04.img base = localtime-m 4
```

Cosmos

Website: <http://gocosmos.org>

Cosmos is an operating system written in C #.

```
Host ~ $ wget http://gocosmos.org/Vault/Cosmos-Ms1.zip
Host ~ $ unzip Cosmos Ms1.zip
Host ~ $ qemu-cdrom Cosmos.iso
```

DexOS

Website: <http://www.dex4u.com>

DexOS is a small program written in assembly language operating system.

```
Host ~ $ wget http://www.dex4u.com/dexiso.zip
Host ~ $ unzip dexiso.zip
Host ~ $ qemu-cdrom dexiso / dexiso.ISO std-vga
```

JNode

Download: http://www.jnode.org/download_latest

JNode is a Java-based operating system. A quick start is at the URL <http://www.jnode.org/node/798> and instructions for operating under KVM is at the URL <http://www.jnode.org/node/2750> available.

```
Host ~ $ wget \
http://prdownloads.sourceforge.net/jnode/jnode-x86-0.2.8.iso.gz
Host ~ $ gunzip jnode x86 0.2.8.iso.gz
Host ~ $ qemu-cdrom jnode-x86-0.2.8.iso std-vga-m 768
```

With KVM, the following options are applied.

```
Host ~ $ kvm-m 768-cdrom jnode-x86-0.2.8.iso-no-acpi \
-No-kvm-net none-vga std
```

thos

Website: <http://www.losethos.com/>

thos is a little loose in C-implemented 64-bit operating system. The shell syntax is based on the programming language C.

```
Host ~ $ wget http://www.losethos.com/LTCD.ISO
Host ~ $ qemu-system-x86_64-cdrom LTCD.ISO-m 512 \
-Net none-soundhw pcspk-boot d
```

MikeOS

Download: <http://mikeos.berlios.de/> # download

Doku: <http://mikeos.berlios.de/handbook.html>

MikeOS is a minimalist operating system. It is similar to DOS.

```
Host ~ $ wget http://download.berlios.de/mikeos/mikeos-2.0.0.tar.gz
Host ~ $ tar xvzf mikeos-2.0.0.tar.gz
Host ~ $ cd ~ mikeos-2.0.0/disk_images
Host ~ $ qemu-cdrom-rtc mikeos.iso base = localtime
```

MonaOS

Website: <http://www.monaos.org>

MonaOS is a simple operating system.

```
Host ~ $ unzip-month 0.3.0alpha9-qemu-0.8.2-windows.zip
Host ~ $ cd ~ mona-0.3.0alpha9-qemu-0.8.2-windows-tap
Host ~ $ qemu-cdrom-fda mona.iso mona.img-boot d-soundhw pcspk
```

SharpOS

Website: <http://www.sharpos.org>

Doku: http://www.sharpos.org/redmine/wiki/3/Howto_Debug_with_QEmu

SharpOS is on. NET technologies based operating system.

```
Host ~ $ tar xjvf SharpOS-0.0.1.tar.bz2
Host ~ $ qemu-fda SharpOS-0.0.1/DiskImages/SharpOS.img
```

Visopsys

Website: <http://www.visopsys.org>

A small operating system with a partitioning tool and other applications.

```
Host ~ $ wget \
http://visopsys.org/files/visopsys/visopsys-0.7_PRE-iso.zip
Host ~ $ unzip-Visopsys 0.7_PRE-iso.zip
Host ~ $ qemu-cdrom-Visopsys 0.7_PRE.iso-m 16
```

Whitix

Website: <http://www.whitix.org/download.php>

Whitix is a simple operating system.

```
Host ~ $ wget http://www.whitix.org/downloads/whitix-current.iso
Host ~ $ qemu-cdrom-whitix current.iso
```

<<< | # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_x86-Architektur/_Exoten "

This page has been accessed 6682 times. This page was last updated on 5 January 2011 at 06:51 clock changed. Content is available under GNU Free Documentation License 1.2

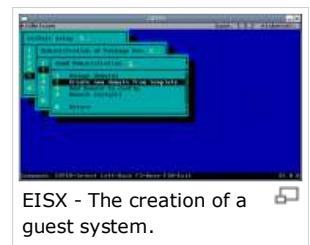
EISX hypervisor Xen-in, metal, Full Virtualization, download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / x86 / hypervisor]])

<<< | # # # | >>> | English

Hypervisor

The hypervisor is on the paravirtualization and full virtualization software layer that lies directly between the hardware and operating systems. Full Virtualization hypervisor require processors with hardware virtualization technologies. To emulate this, requires the host system AMD processors. The kernel module *kvm-amd* is *nested = 1* option to activate the (see <http://qemu-buch.de/d/Installation>). The *CPU-definition* *qemu64 cpu, + svm* be 64-bit processors with hardware virtualization technology emulates.



EISX

Download: <http://www.eisxen.org>
<http://www.eisfair.org/home/download/>
 Recommendation: Standard PC (default), > 256 MB RAM, Hard Drive: 10 GB

The software Xen is a hypervisor that is being developed at the University of Cambridge. Xen is to install complicated and expensive. First, a boot loader is needed to start in several steps in the right order and the necessary software components. It must first of the hypervisor and then the customized operating system for the privileged virtual machine to load. This complex configuration is on EISX not necessary. Based on the proven *Linux eisfair minimal distribution* to install a Xen server in minutes, and virtual machines can be easily set up themselves. The Linux distribution is used here as *eisfair* privileged virtual machine. The size of the disk depends on the planned number of guest systems. It should be at least 10 GB.

```
Host ~ $ qemu-img create-f qcow2 eisxen hd.img-10G
```

To install you start QEMU, or the Kernel-based Virtual Machine with the hard disk and the downloaded CD image. Here are the options *kvm-no-kqemu-or no-state*, since neither of these KQEMU still KVM virtualization software within the authority will comply with.

```
Host ~ $ qemu-hda-eisxen hd.img-cdrom-eisxen rc5.iso \
-Boot d-no-kvm
```

First, ask if the entire hard disk to use *hda*.

```
Use hda for installation (y / n) [yes] [Enter]
```

In test mode is no extra partition for the data necessary.

```
Create extra data partition / data (y / n)? N
Create partitions as listed above (y / n) y
```

After the security check will begin the installation. In the end, after the passwords for the *root* user and asked *ice*. Users of the *ice* is used to configure the system by using menus.

```
Setting password for user root: Enter new UNIX password: ***** Setting password for user eis: Enter new UNIX passw
```

The network configuration is done via DHCP.

```
Use DHCP for network card? (Y / n) [yes] [Enter]
```

After installation, the virtual machine without a CD, but start with more RAM.

```
Host ~ $ qemu-hda-eisxen hd.img-m 512-no-kvm
```

To configure the system log into *ice* as a user. It automatically starts the configuration program. It is recommended to install and configure the virtual machine shut down and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b-eisxen hd.img qcow2 eisxen-f-hd-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-hda eisxen-hd-01.ovl-m 512-no-kvm
```

Under the Kernel-based Virtual Machine use the *option-no-kvm*.

```
Host ~ $ kvm-hda eisxen-hd-01.ovl-m 512-no-kvm
```

can EISX In guest systems can be generated easily from templates. To create a host system from this template you select the menu item *Administration Xen, domU Administration, Create new domain from template*. The package includes templates for the Linux distribution are eisfair. For the first test you take over the guidelines:

```
eisfair1.tar.bz2
Name of domain [eisfair1]: [Enter]
```

Disk size in MB [512]: [**Enter**]
 Memory size in MB [64]: [**Enter**]
 Swap space in MB (0 = no swap) [128]: [**Enter**]
 Power on when domain 0 boots (y / n) [yes]: [**Enter**]

It is the template select (*eisfair1.tar.bz2*). Then a name for the guest system is to forgive. More details are the size of the virtual disk, the RAM and the size of the swap area. EISX installed the new system and submit the generated image in the folder */data/xen/images/down*. Then the network is configured.

Please enter a host name [eisfair1]? **Horst**
 Please enter an IP Address? **10.0.2.20**
 Please enter the subnet mask [255.255.255.0]: [**Enter**]
 Please enter the default gateway? **10.0.2.2**
 Please enter the Doamin [lan.home]: [**Enter**]
 Please enter the DNS? **10.0.2.3**

For the guest system user credentials for the *root* obtained. Guest systems are the menu *Xen administration, domU administration, management domains* managed. This is the guest system (domain) input.

List Xen domains: no name order booted 1-50 no horst Enter command ([1] to manage the domain, Enter = Return): **1**

It will start the guest system.

Please enter command:
 [B] oot,
 b [o] ot & connect,
 [N] ew boot order
 [D] elete
 Enter = Return: **o**

The guest system will boot and you will see the login screen. Will leave this console using the key combination [Ctrl]+[**]**. It is possible to generate additional guest systems with the available templates. Furthermore, you can move their own templates. How EISX find on the Web site. A QEMU image with EISX and a template for generating virtual machines with Ubuntu 6.06 LTS server at the URL <http://www.oszoo.org/wiki/index.php/EisXEN-beta2a-with-ubuntu-6.06.1-server-template> to be downloaded.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_x86-Architektur/_Hypervisor "

This page has been accessed 4670 times. This page was last updated on 26 January 2011 at 16:34 clock changed. Content is available under GNU Free Documentation License 1.2 .

SPARC, Debian, Linux, download iso installation, SPARCstation, server

(Link to this page as [[QEMU-KVM-Buch / guest systems / SPARC Architecture]])

<<< | # # # | >>> | English

SPARC

The emulation of the 32-bit SPARC architecture allows the command *qemu-system-sparc*. This makes it possible to create virtual machines that contain as a basis of available processors from the SPARC family.

Debian GNU / Linux 4.0 Etch

Download: <http://www.debian.org/CD/netinst/>

Recommendation: SPARC architecture, 128 MB RAM, hard disk: from 2 GB.

Debian GNU / Linux is a free Linux distribution that is compiled exclusively from free software. Debian is a reliable server operating system and serves as a stable foundation for many other Linux distributions such as Ubuntu. Debian is available for eleven processor architectures. For example, here is the SPARC platform (32-bit). A virtual hard disk size of 2 GB is sufficient.

```
Host ~ $ qemu-img create-f qcow2 DebianSPARC.img 2G
```

From the download URL to Load CD-image for *sparc* in the section down the *business card image*. Then you start *qemu-system-sparc*.

```
Host ~ $ qemu-system-sparc-hda DebianSPARC.img \
-Cdrom debian-40r3-sparc-boot d-businesscard.iso
```

The installation is similar to the Ubuntu server. Once installed, these options are used.

```
Host ~ $ qemu-system-sparc-hda DebianSPARC.img
```

After installation and configuration is recommended to shut down the virtual machine and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b DebianSPARC.img \
Debian-sparc-f qcow2 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-system-sparc-hda debian-sparc 01.ovl
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_SPARC-Architektur "

This page has been accessed 4004 times. This page was last updated on 27 September 2010 at 06:31 clock changed. Content is available under GNU Free Documentation License 1.2 .

ARM processor architecture, arm processor, qemu-system-arm, Linux, download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / ARM processor architecture]])

<<< | # # # | >>> | English

ARM processor architecture

The emulation of the ARM processor architecture allows the command *qemu-system-arm*.

Linux kernel with a small file system

Download: <http://wiki.qemu.org/Download>

Recommendation: ARM processor architecture, 128 MB RAM.

On the website of QEMU is located in the download section a small test system for the ARM processor architecture. It includes a Linux kernel and a small file system. It downloads the tarball and unpacked it. Linux Example:

```
Host ~ $ wget http://wiki.qemu.org/download/arm-test-0.2.tar.gz host ~ $ tar xzvf arm-test-0.2.tar.gz ~ $ cd host arm-test
```

After switching to the new directory is called QEMU with the external Linux kernel and initial RAM disk. There are two variants. In the first variant QEMU is open as usual in a window.

```
Host ~ $ qemu-system-arm-kernel zImage.integrator \
Initrd-arm_root.img
```

In the second variant, the output in the console of the host system. With the *append-kernel* a boot option passed is: *console = ttyAMA0* redirects output to the console on. *The-Oceanographic* causes not have QEMU window opens.

```
Host ~ $ qemu-system-arm-Oceanographic \
-Kernel zImage.integrator \
Initrd-arm_root.img \
-Oceanographic-append "console = ttyAMA0"
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_ARM-Prozessorarchitektur "

This page has been accessed 5119 times. This page was last updated on 27 September 2010 at 06:32 clock changed. Content is available under GNU Free Documentation License 1.2 .

MIPS processor architecture, processor mips, mips architecture, qemu-system-mips, Debian linux download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / MIPS processor architecture]])

<<< | # # # | >>> | English

MIPS processor architecture

The emulation of the MIPS architecture allows the command `qemu-system-mips`.

Debian GNU / Linux 4.0 Etch

Download: <http://ftp.de.debian.org/debian/dists/etch/main/installer-mips/current/images/qemu/netboot/>

Recommendation: MIPS architecture (R4000), 128 MB RAM, hard disk: from 2 GB.

As an example, the MIPS R4000 processor platform is used with the. This is the platform compiled Linux kernel and `initrd` file for Debian Etch installer from the download. Furthermore, a virtual disk is created for the installation.

```
Host ~ $ qemu-img create-f qcow2 DebianEtch_on_MIPS.img 2G
```

To install the operating system is `mips qemu-system-started` with the following options:

```
Host ~ $ qemu-system-mips-hda-DebianEtch_on_MIPS.img Oceanographic \
  -Kernel vmlinux-2.6.18-4-qemu \
  -Initrd initrd.gz \
  -Append "root = / dev / ram console = ttys0"
```

Here is the ability of QEMU used externally to boot a Linux kernel without boot media must be used. `Root = / dev / ram` devices file is the where the root lies. Furthermore, `console = ttys0` output to the console redirected to. `The-Oceanographic` causes not have QEMU window opens. The output is in the console of the host system was started in the QEMU.

The installation is similar to the Ubuntu server. At the end of the installation a warning message that the bootloader could not be installed. A boot loader is here but not required because the Linux kernel will run directly from QEMU. This message can be ignored. After you install QEMU, discontinue use and start with the following new options:

```
Host ~ $ qemu-system-mips-hda DebianEtch_on_MIPS.img \-kernel vmlinux-2.6.18-4-qemu-initrd initrd.gz \-append "root = / dev/hd1 console = ttys0" Ocea
```

After installation and configuration is recommended to shut down the virtual machine and one or more overlay files to create in order to protect the base image to make changes.

```
Host ~ $ qemu-img create-b DebianEtch_on_MIPS.img \
  -F-qcow2 DebianEtch_on_MIPS 01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-system-mips-hda-DebianEtch_on_MIPS 01.ovl \
  -Kernel vmlinux-2.6.18-4-qemu-initrd initrd.gz \
  -Append "root = / dev/hd1 console = ttys0" Oceanographic
```

<<< | # # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_MIPS-Prozessorarchitektur"

This page has been accessed 3857 times. This page was last updated on 27 September 2010 at 06:32 clock changed. Content is available under GNU Free Documentation License 1.2 .

Coldfire processor architecture, Freescale ColdFire, qemu-system-m68k, linux uclinux download iso installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / Coldfire processor architecture]])

<<< | # # # | >>> | English

Cold Fire architecture

The emulation of the Coldfire architecture allows the command *qemu-system-m68k*. This emulator allows you to boot a uLinux kernel.

Linux kernel with a small file system

Download: <http://wiki.qemu.org/Download>

Recommendation: Coldfire processor architecture, 128 MB RAM.

On the website of QEMU is located in the download section is a small test system for the Coldfire processor architecture. It includes a Linux kernel and a small file system. It downloads the tarball and unpacked it. Linux Example:

```
Host ~ $ wget http://wiki.qemu.org/download/coldfire-test-0.1.tar.bz2
Host ~ $ tar xjvf ColdFire-test-0.1.tar.bz2
Host $ cd ~ cold-fire test-0.1
```

After switching to the new directory is called QEMU.

```
Host ~ $ qemu-system-m68k-kernel vmlinux-2.6.21-uc0-Oceanographic
```

It opens not have a QEMU window. The output is in the console was launched in QEMU.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_Coldfire-Prozessorarchitektur "

This page has been accessed 3207 times. This page was last updated on 7 March 2010 at 08:52 clock changed. Content is available under GNU Free Documentation License 1.2 .

QEMU PowerPC architecture qemu-system-ppc qemu-system-ppc64, Debian Linux iso download installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / PowerPC]])

<<< | # # # | >>> | English

PowerPC

The emulation of the PowerPC architecture commands allow *qemu-system-ppc* *qemu-system-ppc64* and *qemu-system-ppcemb*.

Debian GNU / Linux 4.0 (Etch)

Download: http://www.oszoo.org/wiki/index.php/Debian_Etch_ppc_28PowerPC%29_-_Qemu_Ready

Recommended: PowerPC, 256 MB RAM, hard disk: from 3 GB.

Debian is available for eleven processor architectures. For example, here is the PPC platform. An installed system can be downloaded from FreeOsZoo and unpacked. Here, the corresponding Unix-/Linux-Befehle:

```
Host ~ $ wget \
http://www.oszoo.org/images/qemu-ppc-debian-etch-image.tar
Host ~ $ tar xvf qemu-ppc-debian-etch-image.tar
Host $ cd qemu-debian-etch-ppc-image
```

Start the virtual machine with the script *run-qemu-ssh*.

```
Host ~ $ . / Run-qemu-ssh
```

After booting, login to the *root* user and password *root*. As configured in the script redirect a port to the SSH port, was a login via SSH is possible.

```
Host ~ $ ssh root @ localhost-p 22 000
```

It is recommended to install and configure the virtual machine shut down and *qcow2* format to convert to.

```
Host ~ $ qemu-img convert-O qcow2 debian-ppc-qemu.qcow \
debian-ppc-qemu.qcow2
```

Furthermore, to create one or more overlay files to protect the base image to make changes.

```
Host ~ $ qemu-img create-b debian-ppc-qemu.qcow2 \
-F qcow2 debian-ppc-qemu-qcow2-01.ovl
```

The start is from the overlay file with the following options:

```
Host ~ $ qemu-system-ppc-rtc base = localtime-M-m 256 \ prep
ZImage.prep-kernel-hda debian-ppc-qemu-qcow2-01.ovl \
-Net nic-net user, hostfwd = tcp::: 22000 -: 22
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_PowerPC-Architektur "

This page has been accessed 4935 times. This page was last updated on 27 September 2010 at 06:33
clock changed. Content is available under GNU Free Documentation License 1.2 .



Debian under
qemu-system-ppc.

SuperH architecture QEMU qemu-system-sh4, qemu-system-sh4eb, SuperH RISC, Linux iso download installation

(Link to this page as [[QEMU-KVM-Buch / guest systems / SuperH architecture]])

<<< | # # # | >>> | English

SuperH architecture

The emulation of the SuperH architecture enable the commands *qemu-system-sh4* and *qemu-system-sh4eb*.

Linux kernel with a small file system

Download: <http://wiki.qemu.org/Download>

Recommendation: SuperH architecture, 128 MB RAM.

On the website of QEMU is located in the download area, a test system (*SH-test-0.2.tar.bz2*) for the SuperH processor architecture. It includes a Linux kernel and a small file system. It downloads the tarball and unpacked it.

```
Host ~ $ wget http://wiki.qemu.org/download/sh-test-0.2.tar.bz2
Host ~ $ tar xjvf sh-test-0.2.tar.bz2
Host ~ $ cd ~ sh-0.2-test
```

After switching to the new directory is called QEMU. The output is in the terminal, was launched in the QEMU.

```
Host ~ $ qemu-system-sh4-M r2d-hda sh-linux-mini.img \
          -Kernel zImage-serial null-serial stdio-Oceanographic
Please press Enter to activate this console.
# Uname-a
Shlinux Linux 2.6.29-07 132-ge2f6d6c # 5 Mon Apr 6 22:45:01 JST 2009 sh4 unknown
```

<<< | # # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Gast-Systeme/_SuperH-Architektur"

This page has been accessed 2673 times. This page was last updated on 7 March 2010 at 08:41 clock changed. Content is available under GNU Free Documentation License 1.2 .

QEMU-KVM-Buch / Notes

For qemu-kvm and libvirt - QEMU + Kernel-based Virtual Machine - Wiki

<<< | # # # | >>> | English

Startup options of QEMU and KVM

QEMU Monitor

qemu-img and kvm-img

qemu and kvm-io-io

qemu-kvm-nbd and nbd

libvirt

Spice

More virtualizer and Emulators

chroot (New root directory as a sandbox under Unix / Linux)

Basilisk II (68k Macintosh Emulator)

Bochs (x86 emulator)

FAUmachine (hardware emulator, virtualization solution)

Gxemul (ARM, MIPS, PowerPC and SuperH-Emulator)

Solaris Zones (virtualization at the OS level)

personal alpha (alpha-Emulator)

VirtualBox (Native Virtualization)

Microsoft Virtual PC (Native Virtualization)

VMware ESXi (type-1 hypervisor)

VMWare Player (desktop virtualisation)

VMware Server (Native Virtualization, type 2 hypervisor)

VMware Workstation (Native Virtualization)

Win4Lin, WinBSD, Win4Solaris

Xen (paravirtualization and full virtualization, Type-1 hypervisor)

Useful Tools

Wine (Wine Is Not an Emulator)

GNU Free Documentation License

<<< | # # # | >>> <http://qemu-buch.de>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang "

This page has been accessed 4179 times. This page was last updated on 5 March 2010 at 06:37 clock changed. Content is available under GNU Free Documentation License 1.2 .

Options QEMU and the Kernel-based Virtual Machine, KVM QEMU short and sweet

(Link to this page as [[QEMU-KVM-book / notes / start options of QEMU and KVM]])

<<< | # # # | >>> | English

Contents

- Add a QEMU and KVM
 - 1.1 Standard Options
 - 1.2 USB Options
 - 1.3 File System Options
 - 1.3.1 File System Devices
 - 1.3.2 Virtual file systems exported
 - 1.4 Display Options
 - 1.5 Network options
 - 1.6 Character Devices
 - 1.7 Bluetooth options
 - 1.8 x86
 - 1.9 Non-x86
 - 1:10 Special Linux boot options
 - 1:11 debugging / Expert options
 - 1:12 options userspace emulation
 - 1.12.1 Standard Options
 - 1.12.2 Debug Options
 - 1.12.3 Environment Variables
 - 1:13 Spice Options
 - 1:14 shortcuts

options of QEMU and KVM

In support of host systems with x86 processor architecture, QEMU or KVM is called with the following commands. *Disk_image* denotes the image of the first IDE hard disk (*disk 0*).

```
Host ~ $ qemu [options] [disk_image]
```

The Kernel-based Virtual Machine is often used with *qemu kvm* instead of the command invoked with. Some distributions call to the *with-system-x86_32 qemu*, *qemu-system-x86_64 qemu* or *kvm-*.

```
Host ~ $ kvm [options] [disk_image]
```

x86 (32 bit):

```
Host ~ $ qemu-system-i386 [options] [disk_image]
```

x86 (64 bit):

```
Host ~ $ qemu-system-x86_64 [options] [disk_image]
```

Standard Options

-H, - help

Display of support.

-Version

It displays the version and then exits.

```
-M machine
```

Selection of the emulated machine. If this option is omitted, or the default *option-M PC*, and a bus today's standard PC with a PCI-emulated. With *M-PC-0.11 pc*, the machine type QEMU version 0.11.0 set by the. With *the-M isapc*), a PC model with the older ISA bus (Industry Standard Architecture emulated. The indication of the possible types of machines is done *with-M?*.

```
CPU model
```

Selection of the emulated CPU. The indication of the possible CPU types is done *with-cpu?*. Additional CPU the CPU type definitions may comma be passed to one, for example-cpu qemu64, + svm. With the CPU definition + svm virtualization techniques are processors with hardware-emulated. That is, the host system can serve as the host system for further KVM instances. This requires that the actual host system via AMD processors. The kernel module *kvm-amd* see is the option with *nested = 1* to enable (http://qemu-buch.de/d/Installation). If a virtualization solution within operated another virtualization solution, it is called *nesting*.

```
\ N-smp [, cores = cores] [, threads = threads] [, sockets = socket]
[, Maxcpus maxcpus =]
```

Specifies the number (*s*) of the emulated CPUs. By default, a CPU is emulated. Up to 255 CPUs for the x86 architecture are supported. For the sparc32 architecture limits Linux host system, the number to four CPUs. For x86 architectures, these options are available. Missing information can be calculated. If values for *cores*, *threads* and *sockets* set, the total number of CPUs are omitted.

```
cores = cores
```

Defines the number of cores per socket.

```
threads = threads
```

Defines the number of threads per core.

```
sockets = socket
```

Defines the number of sockets.

```
maxcpus = maxcpus
```

Defines the maximum number of hot-pluggable CPUs.

```
Numa-node [, mem = size] [, cpu = cpu [-cpu]] [, nodeid = node]
```

Simulates a multi-node NUMA system. NUMA (Non-Uniform Memory Architecture) is a memory architecture for multi-processor systems. Each processor has a separate, local memory. Other processors in the system is made possible through a common address space, direct access to this memory (Distributed Shared Memory). NUMA architectures are implemented such as AMD Opteron multi-processor systems on base. In simplified terms, a NUMA node, a memory region in which each byte the same distance (hops) to each CPU (see http://lse.sourceforge.net/numa/faq/). The QEMU monitor informs the command *info* about the NUMA *NUMA* node.

```
mem = size
```

Defines the size of the memory. The default value is 128 MB. If this option is not specified, the resources are shared equally.

```
cpu = cpu [-cpu]
```

Addresses the CPU (s) for configuration. If this option is not specified, the resources are shared equally.

```
nodeid = node
```

Defines the Node ID.

```
-fda/-fdb file
```

The file specified *file* serves as a disk image for drive A or B. On Linux, the device file */dev/fd0* be used for *file*.

```
-Hda file  
-Hdb file  
-Hdc file  
-Hdd file
```

The specified file each *file* is used as a disk image. It is also possible for real hard-disks. However, should access the read-only (*snapshot*) take place. This would cause data loss. On Linux, to be the devices, such as */dev/hda* or */dev/sda*, given to. On Microsoft Windows is a real drive to *\.\Physical drive N*. Where *N* is the drive number (*0* is the first drive).

```
-Cdrom file
```

The specified file is used as an image *file* for the CD-ROM drive (IDE1 Master). It is not possible to *use-hdc and-cdrom* at the same time. Linux and MacOS X can also real CD-ROM drive, */dev/cdrom*, for example *file* used to be. On Microsoft Windows is to the real CD-ROM drive with *\Physical drive D*.

```
-Drive [file = file] [, if = type] [, bus = n] [, unit = m] [, media = d] [, index = i]  
[, Cyls = c, heads = h, secs = s [, trans = t]] [snapshot = on | off] \  
[, Cache = write through | writeback | unsafe | none] [, format = f] \  
[, Serial = s] [, addr = A] [, id = name] [, aio threads = | native] \  
[, Readonly = on | off]
```

Defines a new drive. Possible options are:

```
file = file
```

This option addresses the Image for this drive. If the file name by a comma, it should be double this.

```
if = type
```

This option defines with which kind of interface the drive is connected. Options are: *ide*, *scsi*, *sd* (Secure Digital Card), *mtd* (on-board flash memory), *floppy*, *pflash* (parallel Flash), *virtio* (paravirtualized block device).

```
bus n = unit = m
```

These options define by specifying the bus number and unit ID as the drive is connected.

```
media = d
```

This option sets the media type (*disk* or *cdrom*).

```
index = i
```

Defines an interface on which connector is connected the drive. With Connectors, the *index* number of the specified.

```
cyls = c, heads = h, secs = s [, trans = t]
```

These options have the same meanings as *in-hdachs*.

```
snapshot = on | off
```

The option can be set *on* or *off* and allows the (de) activation of the snapshot function for the

specified drive (see *option-snapshot*).

`cache = none | write through | writeback | unsafe`

The *cache* option can be *none*, *write back*, *write through* (default) or *unsafe* to be set. It controls the use of the cache for the specified drive. With the contact of the parameter *write through* write requests, the host cache for read and used. The confirmation of the write operation is sent to the host system only when the storage subsystem has confirmed the writing process. In the *write back* option is already writing the confirmation sent when the data of the host are stored in the cache. Crashes the host, it can cause loss of data. With *unsafe* contents of the cache never written to the disk. For problems with the host thereby threatening loss of data. At the start *snapshot default* option is *unsafe* as a set. When *none* option cache, the host is not used. QEMU and KVM can cache the data internally. The performance can be certain drivers in combination with the option to *write through* and the image size be less *qcow2*. If more emphasis on speed to put as security, is to apply the *write back* option.

`format = f`

Specifies the image format (eg *format = raw*).

`serial = s`

Defines the number for the allocation of the device.

`addr = A`

Defines the PCI device address of the controller (*virtio*).

`boot = on | off`

If the option is set *boot = on*, boot the drive is made possible by.

`aio threads = | native`

Allows you to choose from *Pthread based disk I / O* and *Native Linux AIO*.

`readonly = on | off`

With *readonly = on* the drive is read-bear.

`Group.id.arg-set = value`

Defines the parameter *arg* for the entry *id* of type *group*. For example:

Drive-set. Id.file \$ = / path / to / image

`-Global driver.property = value`

Defines a global default value for a driver property.

`Mtdblock-file`

The file specified *file* serves as an on-board flash memory image.

`Sd-file`

The file specified *file* serves as a Secure Digital card image.

`Pflash-file`

The file specified *file* serves as a parallel flash image.

`-Boot [order = drives] [, drives once =] [, menu = on | off]`

Defines start of what device or operating system is the Image (x86 PC). The original *format drives* will

boot versions no longer supported in future QEMU.

```
a
Disk
c
Hard drive (default)
d
CD / DVD-ROM
n
Etherboot (1 NIC)
o
Etherboot (2nd NIC)
p
Etherboot (3rd network card)
order = drives
```

Defines the boot sequence. The *option-boot order = adc* will first try to boot from the floppy disk. This is not possible, it tries to CD / DVD-ROM to boot. Should this fail, start the boot from the hard disk.

once = drives

It is only attempted when first starting to boot from this device.

menu = on | off

(De) Activates a menu to select the boot devices. This menu is called in the starting instance using the [F12].

-Snapshot

Changes are not written to the storage media, but in temporary files. With the command *savevm* in QEMU monitor these changes can be saved.

-M megs [M | G]

Defines the amount of memory in megabytes (*megs*). The default value is 128 MB. Optionally, or *G*, the size in megabytes or gigabytes are indicated with *M*.

K-language

Sets the keyboard layout. The specification of the keyboard layout is usually not necessary since it is assumed by the host system. If this is not possible, as with some X servers or VNC, is the attitude of the layout to make this option. In these cases, *en-us* is usually the default.

Symbol Language

ar	Arabic
as	Danish
de	German
de-ch	German (Switzerland)

en-gb	English (UK)
en-us	English (U.S.)
it	Hispanic
et	Estonian
fi	Finnish
so	Faroese
Fri	French
fr-be	French (Belgium)
fr-ca	French (Canada)
fr-ch	French (Switzerland)
hr	Croatian
hu	Hungarian
is	Icelandic
it	Italian
yes	Japanese
lt	Lithuanian
lv	Latvian
mk	Macedonian
nl	Dutch
nl-be	Dutch (Belgium)
no	Norwegian
pl	Polish
pt	Portuguese
pt-br	Portuguese (Brazil)
ru	Russian
sl	Slovenian
sv	Swedish
th	Thai
tr	Turkish

-Audio-help

Prints the list of audio devices with their options.

```
-Soundhw card1 [, ,...] card2
Soundhw-all
```

Enables the audio support for the specified sound cards. Several sound cards are separated by commas.

-Soundhw?

Indication of the supported sound cards.

-Soundhw pcspk

Emulates the PC speaker.

Soundhw-sb16

Emulates the sound card Creative Sound Blaster 16th

Soundhw-es1370

Emulates the sound card ENSONIQ AudioPCI ES1370.

Soundhw-ac97

Emulates the sound card Intel 82801AA AC97 audio. In the Linux guest system requires the kernel module for AC97 *i810_audio* the following option: *modprobe i810_audio clocking = 48000*

-Soundhw pcspk, sb16

Emulates the PC speakers and the Creative Sound Blaster 16th

Soundhw-all

Enables all supported sound cards.

-Device driver [, prop [= value][,...]]

Adds the device *driver* added. *With-device?* Get a list of possible devices.

prop = value

Defines the properties of the device. *With-device driver?* Get a list of possible properties for the specified device.

Adds a device added. *With-device?* Get a list of possible devices.

-Name string1 [, string2 process =]

Defines a name for the instance. This name appears in the title bar of the SDL window. Furthermore, this name for the VNC server is used. *String2* with the process name for the *top command* established under Linux.

Uuid-% 08x-% 04x-% 04x-% 04x-% 012x

Defines a UUID (Universally Unique Identifier) for the instance.

USB options

QEMU emulates a PCI UHCI USB controller. This allows USB devices to virtual and real USB devices to the host system (Linux only) to connect to the virtual machine. QEMU automatically creates if necessary virtual USB hub.

Usb

Enables the USB support.

USBDevice-name

Adds an additional USB device. Alternatively, QEMU monitor the command *usb_add* used. Possible USB devices are:

mouse

This option enables a virtual USB mouse, overriding the PS/2-Maus-Emulation.

tablet

With this option, the mouse pointer is used in both the guest and the host system. The mouse pointer leaves the window of the host system is automatically switched to the hands of the host system. This option overrides the PS/2-Maus-Emulation.

disk: [format = format] file

With this option) is a mass storage (USB stick virtual emulated. *File* called the image of the storage media. *Format* is the image format (eg *format = raw*).

host: bus.addr

With this option, a USB device to the host system through its bus address is directly addressed. This is only possible with Linux and still in the experimental stage.

host: vendor_id: product_id

With this option, a USB device to the host system is through its vendor and product ID directly addressed. This is only possible with Linux and still experimental.

wacom tablet

Emulates a virtual PenPartner Wacom Tablet. the possibilities of the *tablet* option addition, the pressure on the pointing device to be evaluated. This requires the Tslib library.

keyboard

This option enables a virtual USB keyboard and overwrites the existing PS/2-Tastatur-Emulation.

serial: [VendorID = vendor_id] [, productid = product_id]: dev

Enables a virtual serial converter at the specified character device *dev* to the host system. Emulate the FTDI chip FT232BM. The specification vendor ID and product ID will override the default values of 0403:6001.

braille

Activates the Braille device. This the BrAPI used to display the Braille edition for the visually impaired or a fake device.

net: options

Enables a virtual network adapter that supports the protocols CDC Ethernet and RNDIS. The options correspond to the *options of-net nic*. For example, a user mode network stack with USB is generated with the following command:

qemu [... OPTIONS ...]-net user, vlan = 0-USBDevice net: vlan = 0

bt [: hci-type]

Enables a Bluetooth dongle. The possible parameters of *hci-type* correspond to those of *hci-bt* (see Bluetooth options). If *hci* type not specified, the default *setting-bt hci, vlan = 0*. This USB Device USB implements the Transport Layer of HCI.

File System Options

00:13 From QEMU, the *options-and-support fsdev virtfs*.

File System Devices

The general syntax for defining file system devices is *fsdev-fstype, id = id [, options]*. Any device with a string *id* must be assigned. This is used for unique identification. Depending on the type of devices (*fstype*), certain options apply.

Fsdev-local, id = id, path = path, security_model = security_model

Specifies a device for a local file system.

local

The *local* file system is only available on Linux.

path

Defines the path to be exported. This information is mandatory.

`security_model`

Defines the security model chosen. This information is mandatory.

Virtual exported file systems

The general syntax for exporting file systems as virtual file systems `virtfs-fstype [, options]`. Depending on the type of devices (*fstype*), certain options apply.

`Virtfs-local, path = path, mount_tag = mount_tag, security_model = security_model`

Exports a local file system as a virtual file system.

`local`

The *local* file system is only available on Linux.

`path`

Defines the path to be exported. This information is mandatory.

`security_model`

Defines the security model chosen. This information is mandatory.

`mount_tag`

tag with the exported file system is mounted to the Custom. This information is mandatory.

Display options

`-Oceanographic`

Disables the graphic output and directs the serial input and output to the console. This can be controlled using special keyboard shortcuts.

`-Curses`

Normally, the Simple DirectMedia Layer (SDL) for the VGA output is used. This option allows the text mode, the VGA output of the host system via the Curses / Ncurses interface. In graphical mode at this option, no output.

`-No-frame`

Disables the window frame and thus the instance is the full screen area for the guest system.

`-Alt-grave`

Uses the [Ctrl] + [Alt] + [Shift] instead of [Ctrl] + [Alt] to release the mouse.

`-Ctrl-grave`

Uses the right [Ctrl] instead of [Ctrl] + [Alt] to release the mouse.

`-No-quit`

Disable the button to close the window of the instance.

`-Sdl`

Enables the Simple DirectMedia Layer (SDL) for the VGA output. This option is with *the-vnc* sense together.

-Portrait

Rotates the graphic output by 90 degrees counter-clockwise (only PXA LCD).

-Vga [std | cirrus | vmware | xenfb | none]

Select the type of the emulated graphics card.

std

Simulate a standard VGA card with VESA Bochs Extensions. Does the host system, the VESA 2.0 VBE extensions, this option for higher resolutions (> = 1280x1024x16) are selected.

cirrus

It emulates a CL-GD5446 PCI VGA card. This option is set as the default. All versions starting from Microsoft Windows 95 recognize this graphics card. For optimal performance in the host and guest system in each set a color depth of 16 bits.

vmware

Simulates a VMware SVGA II-compatible adapter. This option is used for guest systems with installed video drivers, the VMware Tools.

xenfb

Xen emulates a frame buffer. For a Linux guest system module, the corresponding command *modprobe* with *xenfb* loaded.

none

Disables the VGA card.

-Std-vga (up QEMU 0.9.1)

Simulate a standard VGA card with VESA Bochs Extensions. By default, a CL-GD5446 PCI VGA card will be emulated. This option was starting QEMU with the option *vga-detached* 0.10.0.

-Full-screen

Starts the emulator in full screen mode.

-Vnc display [, option [, option [,...]]]

Starts the instance to a VNC server on display number *display*. Normally used for VGA output, the Simple DirectMedia Layer (SDL). With *the-vnc* it is possible to output through a VNC session redirect. Useful it is to apply the *option-USBDevice tablet*. Furthermore, the keyboard layout with *the-k* are given. To *display* syntax is possible:

host: d

TCP connections are allowed from the host to display *d* only. The TCP port is 5900 + *d*. The *host* parameter can be omitted if the server to connect to all the interfaces.

unix: path

The connection is made possible by a UNIX domain socket. The *path* parameter defines the path to the socket.

none

VNC is initialized but not started. VNC in QEMU monitor is started with the command *change vnc*.

The options are presented in order of *display*. Multiple options are separated with commas.

reverse

The connection to the VNC client is reverse. In a network connection *d* is a TCP port number and no display number.

password

Required for the connection to the client a password-authentication. The password must be in the QEMU monitor with the command *change vnc password* to be set.

tls

To increase security, is used for the connection from client to server anonymous Transport Layer Security (TLS). This option should the option be combined with *x509* or *x509verify*.

x509 = / path / to / certificate / dir

This option is the option to use *tls* together and requires a *x509* certificate to the secured connection. The server will then transmit its *x509* certificate to the client. The definition of a password is recommended. After the equals sign specifies the path to the certificate file.

x509verify = / path / to / dir certificate

This option is the option to use *tls* together and requires *x509* certificates for safe connection. The server will then transmit its *x509* certificate to the client and calls from the client to the *x509* certificate. Then the server checks the client certificate against the CA certificate. After the equals sign specifies the path to the certificate file. The definition of a password is recommended.

sasl

The option requires the use of the Simple Authentication and Security Layer (SASL) for authentication of the VNC client to the VNC server. The SASL framework provides the application protocol is a standard way of negotiating communication parameters between the server and client. It can be used including authentication methods, PAM, GSSAPI / Kerberos, LDAP, SQL database and one-time keys. By configuring the SASL implementations, a method is defined. Was QEMU / KVM support for SASL compiled, this is done through the SASL configuration file / *etc/sasl2/qemu.conf*. Alternatively, a different configuration file with the variable *SASL_CONF_PATH* referenced. It is recommended that SASL is always used in conjunction with TSL and *x509*. Example:

qemu [... OPTIONS ...]-vnc: 1, sasl-monitor stdio

acl

Was QEMU / KVM with support for ACL compiled, this option enables the access control of the *x509* client certificate and the SASL against an ACL (Access Control List). In *x509* client certificates is testing the Distinguished Name. This, for example, the form *C = DE, O = BLA, = London, CN = Robert L*. When the SASL user name is checked. Depending on the type SASL plugin, the user in the forms such as *robert* or appear *robert@beispiel.de*. If the *acl* flag is set, the initial access control list is empty and only has the Deny policy. This prevents anyone from gaining unauthorized access, as long as the VNC server has not loaded any ACL. To manage the access control list are the commands *acl_show*, *acl_policy*, *acl_allow* *acl_remove* and the QEMU monitor.

lossy

Allows a lossy compression method (gradient, JPEG, ...). If this option is enabled, there may be lossy updates the screen layout. It will be taking a much lower bandwidth.

Network Options

-Net nic [, vlan = n] [, macaddr mac =] [, model = type] [, name = st]
[, Addr = str] [, vectors = v]

Creates a new network card. If the *option-net* does not use a single network card is emulated.

vlan = n

The NIC is the VLAN associated with *n* (default *n* = 0).

macaddr = mac

Macaddr with a MAC address is specified.

model = type

parameter *model* with can be the network cards *virtio*, *i82551*, *i82557b*, *i82559er*, *ne2k_pci* (default for x86 PC with older QEMU versions), *ne2k_isa*, *PCnet*, *rtl8139*, *e1000* (default for x86-PC), *smc91c111*, *lance* and *mcf_fec*. Setting It does not support all network cards from all host systems. With-net nic, *model*=? Can the available network card list.

name = str

The network card is named for better addressing. The QEMU monitor *info* with this name is displayed *network*.

addr = str

Defines the PCI Device Address.

vectors v =

Optionally, set for PCI network cards the number of MSI-X vectors (Message Signaled Interrupts). This specification has currently no effect on virtio cards. With a value of 0 MSI-X is disabled.

```
-Net user [, vlan = n] [, name = str] [, net = addr [/ mask]] [, host = addr] \
[, RESTRICT = y | n] [, hostname = host] [, dhcpstart = addr] [, dns = addr] \
[, Tftp = dir] [, boot file = f] [, hostfwd = rule] [, guestfwd = rule] \
[, Smb = dir [, smbserver = addr]]
```

Connects the user-mode network stack to VLAN *s*. This does not have administrator privileges are required. QEMU monitor settings are displayed in the *info network*.

vlan = n

The user-mode stack is the VLAN associated with *n* (default *n* = 0).

name = str

A user-mode stack is named for better addressing. The QEMU monitor *info* with this name is displayed *network*.

net = addr [/ mask]

Defines the network address and optional netmask for the host system (Default = 10.0.2.0 / 8).

host = addr

Defines the visible to the host system IP address of the host system. By default, this is the second IP address in the guest network, for example xxx2.

restrict = y | yes | n | no

If this option is enabled, the host system is isolated. That is, the guest system can not establish a network connection to the host and the routing through the host is not possible. This option does not affect the forward rules (parameters and *hostfwd* *guestfwd*).

hostname = host

Defined by the integrated DHCP server assigned DNS name for the guest.

`dhcpstart = addr`

Defines the first address of the IP range can assign the DHCP server. This area includes 16 IP addresses. By default, this region lies between xxx16 xxx31 and. For example, with `dhcpstart = 10.0.2.50` DHCP to the host system an IP address from 10.0.2.50 given by.

`dns = addr`

Defines the visible to the host system IP address of the integrated DNS server. By default, this is the third IP address in the guest network, for example xxx3. This address does not match the IP address of the host system.

`tftp = dir`

Allows the host system via an integrated TFTP server read access to files on the host system, the path begins with *you*. The TFTP client on the host system must be configured in binary mode. The host IP address is 10.0.2.2.

`boot-file = file`

When using the user mode network stack, this option with the *file* specified as a BOOTP file published in the network. In conjunction with the *tftp* host system, a local directory of the host system started to be one of. Such as the use of *pxelinux*:

`qemu-hda-boot linux.img n-net user, tftp = / path / to / tftp / files, boot-file = / pxelinux.0`

`smb = dir [, smbserver = addr]`

Enables data exchange between host and guest system via Samba share (windows share) of the list *you* in the host system. This is a Samba server requires on the host system. Was on the host system is already installed Samba server, and configured, the guest system shares with the URL \\ 10.0.2.4 \\ be accessed on the. The user, under the QEMU / KVM is running must have the rights to start this Samba server. The parameter *smbserver* defines the IP address of the Samba server. By default, this is the fourth IP address in the guest network, for example xxx4. In the host system to have a Samba server / *sbin* / *smbd* be installed *usr* / under.

`hostfwd = [tcp | udp]: [hostaddr]: host port [guestaddr]: guest-port`

Allows the redirection of TCP or UDP connections from a port of the host system to a port of the host system. This option can be specified multiple times. If neither *tcp* or *udp* as the connection type specified, the TCP protocol is used. By specifying *hostaddr*, the forward rule defined in the host network interface bound to be one. If not specified *guestaddr*, whose address is the first value from the internal DHCP server default IP assigned (xxx15). The QEMU monitor informs the command *info* on the *Usenet* defined forward rules. Furthermore, the QEMU monitor rules with the new forward command will be added *hostfwd_add*. The command *hostfwd_remove* rules Forward deleted.

`guestfwd = [tcp]: server: port-dev`

Initiates a TCP connection from the host system to the IP address (*server*) and port (*port*) to a character device (*dev*) of the host system in order. This option can be specified multiple times.

```
-Net tap [, vlan = n] [, name = name] [, fd = h] [, ifname = name] \
[, Script = file] [, script = dfile down] [, sndbuf = nbytes] [, vnet_hdr = on | off] \
[, Vhost = on | off] [, vhostfd = h]
```

TAP connects the network interface of the host system with the VLAN s.

`script = file`

It is network-script *file* (default = / etc / *qemu-ifup*) used to activate it.

`down script = dfile`

It is *dfile* script (default = / etc / *qemu-ifdown*) used to disable it. Where the performance of the

scripts are prevented, is *no script = script* or *down* to set *no =*.

```
fd = h
```

Interface with *fd* is a handle to an existing TAP network specified.

```
sndbuf = nbytes
```

With the option *sndbuf* limit the size of the send buffer. The default value of *sndbuf = 1048576* can be deactivated with *sndbuf = 0*.

```
vnet_hdr = on | off
```

If the option *vnet_hdr* = set *off*, *IFF_VNET_HDR*, the TUN / TAP-flag disabled. This flag allows you to send and receive large packets or packets with partial checksums. With poor support for *IFF_VNET_HDR* this option should be disabled.

```
vhost = on
```

Enabled the experimental kernel accelerator.

```
vhostfd = h
```

Connects to an already open network device *hours* ago.

```
-Net socket [, vlan = n] [, name = name] [, fd = h] [, listen = [host]: port] \
[, Connect = host: port]
```

VLAN connects the *n* with a different VLAN on a TCP socket connection. Indicated *listen*, instance waits for incoming connections on the specified port (*host* is optional.) In the second instance, connects you to *connect* with the first instance. With *fd* is a handle of an existing TCP socket can be specified.

```
-Net socket [, vlan = n] [, name = name] [, fd = h] [, mcast = Maddr: port]
```

Creates the VLAN *n*, with the same multicast *Maddr* Venue Address and Port *port* can be used together with other instances. Multiple instances can run on different hosts and use the same bus. The multicast support is compatible with User Mode Linux (*eth n = mcast*). With *fd* is a handle to an existing multicast UDP sockets can be specified.

```
-Net vde [, vlan = n] [, name = name] [, sock = socketpath] [, port = n] \
[, Group = groupname] [mode = octalmode]
```

Was QEMU / KVM with support for VDE is compiled with this option, *n* VLAN to the port *n* of a VDE switches (Virtual Distributed Ethernet). This virtual switch is installed on the host system and waits for incoming connections on the socket *socketpath*. With the group *groupname* *octalmode* and the mode it is possible to adjust ownership and permissions.

```
-Net channel, port: dev is (replaced by-net user, guestfwd =...)
```

Directs the network connection to the port character device to *dev*.

```
Net-dump [, vlan = n] [file = f] [, len = n]
```

Stores the data of the network traffic from VLAN *s* in the *f* file (Default = *qemu-vlan0.pcap*). It can with *n* the maximum number of bytes per packet is set (default = 64k). The file format is used *libpcap*. This allows the data with the tools to analyze *tcpdump* or Wireshark.

```
-Net none
```

Disable all network devices. If, however, not given the *option-net*, is the default *Optionnnn-net nic-net user*.

```
Netdev-[user | tap | vde | socket], id = str [, option] [, option ][,...]
```

QEMU with 0.12.0, the *option* to define virtual networks introduced *netdev*. Depending on the type of virtual network (*user*, *tap*, *vde*, *socket*) are certain parameters apply. must be a string that uniquely identifies *id* assigned to this case.

Tftp-dir is (replaced by-net user, tftp =...)

Allows the host system via TFTP server read access to files on the host system whose path begins with *you* one. This option sets the *pre-net user*. The TFTP client on the host system must be configured in binary mode. The host IP address is 10.0.2.2.

Bootp-file is (replaced by-net user boot file =...)

When using the user mode network stack, this option with the *file* specified as a BOOTP file published in the network. In conjunction with the *tftp-host* system, a local directory of the host system started to be one of.

-Dir smb (to be replaced by-net user, smb =...)

Enables data exchange between host and guest system via Samba share (windows share) of the list *you* in the host system. This is a Samba server requires the host system. The user, under the QEMU / KVM is running must have the rights to start this Samba server. Furthermore, the *user-net* option provided.

-Redir [tcp | udp]: host-port: [guest-host]: guest-port (to be replaced by-net user, hos

Allows the redirection of TCP or UDP connections from a port of the host system to a port of the host system. This option sets the *pre-net user*. This option can be specified multiple times. This option is replaced with-net user, hostfwd.

Character Devices

The general syntax for defining character devices-is *chardev backend*, *id = id [, options]*. Depending on the type of devices (*backend*) options are certain to apply. Any device with a string *id* must be assigned. This is used for unique identification. *The-chardev* can be used multiple times.

mux = on | off

With *mux = on* mode, the multiplexing enabled. This means that the character device of multiple front-ends are used. With the key combination [Ctrl] [A] and [C] of the input focus between the connected front-end is changed.

Chardev-null, id = id [, mux = on | off]

With *null* device is a blank created. It does not send data and discarding received data.

```
Chardev-socket, id = id [, host = host], port = host [, to = to] \
[IPv4], [ipv6] [, nodelay] [, server] [, nowait] [, telnet] \
[, Mux = on | off]
```

With the option *socket* socket is a two-way created. In this case it is a TCP socket, as defined *host*.

host

For a socket in listen mode, optionally, the host address is specified. a socket in the transmit mode, for *host* with the address of the remote host specified. Will *host* defined, the value is used by *Default-0.0.0.0*.

port

For a socket in listen mode is specified with this option the port. a socket in the transmission mode for the port to *port* of the remote host specified. can either *port* number or service name specified are as a port.

to

This option is relevant only for sockets in listen mode. If this option is set and can not connect to *port* to be prepared, QEMU will have a higher port to connect. With *to* port is the highest given.

ipv4, ipv6

Defines the protocol to use: IPv4 or IPv6. If no protocol is defined, any protocol.

nodelay

Disables the Nagle algorithm.

```
Chardev-socket, id = id [, server] [, nowait] [, telnet] \
[, Mux = on | off]
```

With the option *socket* socket is a two-way created. In this case it is a Unix socket, as defined *path*.

path

Defines the local path to the socket.

server

Defines a connection on the waiting socket (list).

nowait

QEMU does not necessarily waiting for a connection of the client on the specified socket.

telnet

It can be interpreted Telnet escape sequences.

```
Chardev-udp, id = id [, host = host], port = port \
[, Localaddr localaddr =] [, local port = local port] [, ipv4], [ipv6] \
[, Mux = on | off]
```

With the option *udp* traffic is received by the host system on a remote host via UDP broadcast to.

host

Defines the remote host to connect to. If there is no host *localhost*.

port

Defines the port on the remote hosts to which it connects to.

localaddr

Defines the local IP address to which it connects to. If no IP address is specified, the value is used by *Default-0.0.0.0*.

local port

Defines the local port to which it connects to. If no port is defined, any port is used.

ipv4, ipv6

Defines the protocol to use: IPv4 or IPv6. If no protocol is defined, any protocol.

-Chardev msmouse, id = id [, mux = on | off]

With the option *msmouse* mouse for the host system, the events of a Microsoft-emulated. The QEMU monitor informs the command *info* on *mice*, the mice.

```
Chardev-vc, id = id [[, width = [width], height = height]] \ 
[[], Cols = cols] [, rows = rows]] [, mux = on | off]
```

With the option *vc* QEMU is a connection to the text console produced. At that text console with the key combination [Ctrl] + switch [Alt] + [2]. Optionally, the size in pixels with *height* and *width* are adjusted. Alternatively, the size of the text console in characters with defining the *cols* and *rows*.

```
Chardev-file, id = id, path = path [, mux = on | off]
```

The option *file* is the traffic from the host system received a file written to. *Path* with the path to the file defined. This file exists, it is created. The contents of an existing file is overwritten.

```
Chardev-pipe, id = id, path = path [, mux = on | off]
```

With the option *pipe* system is a two-way connection to the host created. On host systems with Microsoft Windows \ is a duplex-pipe with the path \ \. *Pipe* used *path*. On other host systems, there are two pipes. In the pipe system *path.in* data written to the guest-submitted. Data from the host system from the pipe *path.out* read. Since QEMU these FIFOs do not own port, the pipes must already exist.

```
Chardev-console, id = id [, mux = on | off]
```

The *console* device sends traffic to host system to the standard output of QEMU. This option is not host systems running Microsoft Windows.

```
Chardev-serial, id = id, path = path [, mux = on | off]
```

The *serial* device connects to a serial device to the host system to her. With *path* system, the corresponding device of the host specified. This option is not host systems running Microsoft Windows.

```
Chardev pty-id = id [, mux = on | off]
```

The device creates a new *pty* pseudo terminal on the host system and connects to it. This option is not host systems running Microsoft Windows.

```
Chardev-stdio, id = id [, signal = on | off] [, mux = on | off]
```

The *stdio* Device connection is a standard input and output QEMU process of her. This option is not host systems running Microsoft Windows. If control signals are activated in the terminal, QEMU can use [Ctrl] + [C] to leave. With *signal = off*, this is prevented. The default is *signal = on*.

```
Chardev-braille, id = id [, mux = on | off]
```

The *tty* device is connected to the server establishes a local BrIAPI.

```
Chardev-tty, id = id, path = path [, mux = on | off]
```

The *tty* device can host systems with Linux, Sun, FreeBSD, NetBSD, OpenBSD and DragonFlyBSD be applied only in and makes connection to the local TTY device produces a. With *path* system, the corresponding device of the host specified.

```
Parport-chardev, id = id, path = path [, mux = on | off]
```

The *parport* device can host systems with Linux, FreeBSD and DragonFlyBSD be applied only and provides for connection to the local parallel port creates a. With *path* system, the corresponding device of the host specified.

Bluetooth Options

```
-Bt hci [...]
```

Defines the functions of the corresponding Bluetooth Host Controller Interface (HCI). The parameters of *bt* are those of the *HCI-host* system-allocated. If in a virtual machine is just a example to emulate HCI, only the first definition given by-*hci* [...] used *bt*. The transport layer is determined by the machine type.

Currently available only from the Nokia N800 and N810 emulated Internet Tablets with a HCI. The following three species are recognized.:

-Bt hci, null

This is the default setting. The corresponding Host Bluetooth HCI has no internal logic and respond to any commands or events.

-Bt hci, host [: id]

The corresponding heads the HCI commands and events to and from the physical HCI named *id* (default = hci0) the host system to continue. The option is only available on systems with BlueZ (Linux Bluetooth protocol stack) is available.

Bt-hci [, vlan = n]

A virtual standard HCI is applied, the Bluetooth scatternet *n* (default = 0) takes part in the. A scatternet is a group of independent and asynchronous piconets. A piconet is a Personal Area Network (PAN), on the viewing devices connected via Bluetooth. As with VLANs can communicate only devices in the same scatternet.

Vhci-bt [, vlan = n]

The option is only available on Linux, and places in HCI Scatternet *n* (default = 0). The HCI is connected to the Bluetooth stack of the host system. This communication between host and guest systems enables a common scatternet. But the Linux driver is VHCI required.

Bt-device: dev [, vlan = n]

Emulates a Bluetooth device *dev* and placed it on the network *n* (default = 0). QEMU can emulate the moment only the following device.

keyboard

Emulates a virtual keyboard with wireless Human Interface Device Protocol (HIDP).

x86

-Win2k-hack

This option is for the installation of Microsoft Windows 2000 requires as a guest operating system since its bow indicates the hard drive during installation as full. After installation, this option is removed.

-Rtc-td-hack (to be replaced by-rtc driftfix)

For problems with time-drift in Microsoft Windows ACPI HAL with this option is used. It is determined how many timer interrupts are not processed by the Windows guest system and attempts to apply it again.

-No-fd-bootchk

It disables the test of signature boot from floppy disks. This may be relevant for diskettes older operating systems.

-No-acpi

Disabled the Advanced Configuration and Power Interface (ACPI).

-No-hpet

Disables support for the High Precision Event Timer (HPET).

Balloon-none

Disables the ballooning device.

```
Balloon-virtio [, addr = st]
```

Enables the ballooning device. This is the default setting. For Linux as host system with the memory ballooning to customize the memory at run time is possible.

```
addr = str
```

Defines the PCI Device Address.

```
-Acpitable [sig = str] [, rev = n] \ [, oem_id = str] [, oem_table_id = st]
[, Oem_rev = n] [, asl_compiler_id = str] [, asl_compiler_rev = n] \ [, data = file1 [: f]
```

Configures the ACPI table with the specified header fields and contents of the files.

```
SMBIOS binary-file =
```

Invites SMBIOS entry of a Binary file. The SMBIOS specification (System Management BIOS) define data structures and access for system management BIOS level. These specifications are defined by the Distributed Management Task Force (DMTF). DMTF is a standards body for IT companies.

```
SMBIOS-type = 0 [, vendor = str] [, version = str] [, date = str] [, release =% d.% d]
```

SMBIOS fields defined by the type 0

```
SMBIOS-type = 1 [, manufacturer = str] [, product = str] [, version = str] \
[, Serial = str] [, uuid = uuid], [sku = str] [, family = str]
```

SMBIOS fields defined by the type 1

non-x86 architecture

The emulators of other processor architectures are called as follows.

ARM architecture:

```
Host ~ $ qemu-system-arm [options] [disk_image]
```

MIPS architecture (32 bit, 64-bit):

```
Host ~ $ qemu-system-mips [options] [disk_image]
Host ~ $ qemu-system-mips64 [options] [disk_image]
```

Mipsel architecture (32 bit, 64-bit):

```
Host ~ $ qemu-system-mipsel [options] [disk_image]
Host ~ $ qemu-system-mips64el [options] [disk_image]
```

PowerPC (32 bit, 64-bit):

```
Host ~ $ qemu-system-ppc [options] [disk_image]
Host ~ $ qemu-system-ppc64 [options] [disk_image]
Host ~ $ qemu-system-ppcemb [options] [disk_image]
```

SPARC architecture:

```
Host ~ $ qemu-system-sparc [options] [disk_image]
```

Coldfire architecture:

```
Host ~ $ qemu-system-m68k [options] [disk_image]
```

ETRAX CRIS architecture:

Host ~ \$ **`qemu-system-cris`** [options] [disk_image]

SuperH architecture:

Host ~ \$ **`qemu-system-sh4`** [options] [disk_image]
 Host ~ \$ **`qemu-system-sh4eb`** [options] [disk_image]

MicroBlaze Architecture:

Host ~ \$ **`qemu-system-MicroBlaze`** [options] [disk_image]

Additional options:

-G WxH [xDEPTH]

Set initial screen resolution and color depth (for *qemu-system-**).

String-prom-env

The SPARC-system emulator also has the option to define a OpenBIOS variables in the NVRAM.

Semi-hosting

Enables Hosting Semi Syscall Emulation (only for ARM and M68K). In the ARM architecture, the Angel interface is implemented. For the M68K architecture implements the ColdFire GDB interface. In this case the host system to directly access the host file system is allowed. This can only be trusted host systems used.

Special Linux boot options

Using these options on a Linux or multi-boot kernel is loaded, without this installed on a virtual hard disk.

-Kernel bzImage

Uses the file *bzImage* as kernel image. The kernel can be either a Linux kernel, or a multi-boot kernel.

Initrd-file

Uses the *file* file as initial ram disk.

-Initrd "arg = foo file1, file2"

This option is only available for multiboot kernels. With *file1* and *file2* are called modules. *Arg = foo* is a parameter for the first module.

-Append cmdline

Cmdline used as a kernel boot options. These options depend on the guest kernel.

Debugging / Expert options

Serial-dev

Directs the serial port to the device file to *dev*. The default is *vc* in graphical mode and *stdio* in text mode. This option can be used several times to emulate up to four serial ports. To disable all serial ports, *none* has the *option* to apply *serial*. Possible device files are:

vc [: WxH]

Defines a virtual console. Optionally, the width and height in pixels (for example, *vc: 800x600*) or the number of characters (for example, *vc: 80Cx24C*) specified.

pty

Under Linux, this option is assigned a pseudo-TTY. A new PTY is automatically allocated.

none

There is no device allocated.

null

The redirect to an empty device.

/ Dev / XXX

On Linux system with this option, the serial port of the host to a tty device of the host as `/dev/ttys0`, redirected to. The parameters correspond to those of the host device.

/ Dev / parportN

On Linux, with this option, the serial port of the host system is redirected to a parallel port N of the host system. For the moment SPP and EPP features are used.

file: filename

Directs the output of the serial port of the host system to a file. A submission is not possible.

stdio

On Unix, this is the standard for the input and output (keyboard, screen).

pipe: filename

Directs the output of the serial port of the host system into a named pipe to.

COMn

Windows Under this option, the serial port of the host system to the serial port n of the host, such as COM1, diverted.

udp: [remote_host]: remote_port [@ [src_ip]: src_port]

This option implements *UDP Net Console*. *remote_host* or the parameters specified *src_ip* If not, the default value is used by 0.0.0.0. If no source port) is defined *src_port* (, a random port number is selected. For a simple console command is used without *netcat*. This KVM is QEMU or with *the-serial udp:: 4555 netcat* with options and *nc-u-l-p 4555* start. The expenditure of the body appear on the Net console session. To send characters via Net-Console, the source port is set. For example-*serial udp:: 4555 @: 4556th*

tcp: [host]: port [, server] [, nowait] [, nodelay]

The TCP Net Console has two modes. A mode is called a *location*. In the other mode, the instance is waiting for a connection from *location*. By default, the TCP-net is sent to the console port of the host. When using the optional *server* expects the instance that the client socket application connects to the port one, it was because *nowait* specified. The option disables the Nagle buffering *nodelay* algorithm. If the option is omitted, *host* address is the IP adopted 0.0.0.0. It is only possible to establish a TCP connection. With a Telnet client connects you to the appropriate device.

telnet: host: port [, server] [, nowait] [, nodelay]

It uses the Telnet protocol. The options have the same functions as *in-serial tcp*. The difference is that the connection to the port similar to the Telnet protocol works. This makes it possible to use the MAGIC_SYSRQ sequence. On Unix telnet client, this sequence with the Ctrl key combination []+[] initiated the command and *send break* and [Enter] performed.

unix: path [, server] [, nowait]

It uses a Unix domain socket. This option works in the same way as the *option-serial tcp* with the exception that the Unix domain socket path for the connection uses.

`mon: dev_string`

This option allows the QEMU monitor to another serial port forward. The monitor is followed accessed using the key combination [Ctrl] + [A] of [C]. In this example, the monitor will be redirected to a telnet server (port 4444):

`-Serial mon: telnet::: 4444, server nowait`

`braille`

Was QEMU / KVM with support *brlapi* compiled, this option enables the Braille device. This the BrIAPI used to display the Braille edition for the visually impaired or a fake device.

`msmouse`

Enables a serial three-button mouse. In the host system, the Microsoft protocol is set.

`Parallel-dev`

Directs the parallel port to char device to *dev*. This option can be used up to three times to simulate multiple ports. With *all-parallel none-parallel* port disabled. On Linux host system, the parallel ports / *dev/parport0* up / *dev/parport2* be addressed directly. You have previously received for the account under which rights QEMU or the Kernel-based Virtual Machine is running, read and write permissions are assigned.

`Monitor [control,] dev`

Redirects the QEMU monitor to the device file to *dev*. The device files are similar to those of the serial ports. By default it is *vc* in graphical mode and *stdio* in text mode. The option enables the QEMU monitor *control* protocol.

`QMP-dev`

Is equivalent *to-monitor control ,....* The QMP communications device file is redirected to the *dev*. QMP (QEMU Machine Protocol) is an on JASON (http://www.json.org based management interface that lets applications communicate with the QEMU instance) allows (see http://www.linux-kvm.org/page / monitor protocol).

`mon-chardev = [name] [mode = readline | control] [, default]`

Defines the monitor chardev *name* on.

`Debugcon-dev`

Directs the expenditure of the debug console to the device file to *dev*. The debug console is an I / O port. It typically uses port is *0xE9*. The possible device files correspond to those of *serial-dev*. The default is *vc* in graphical mode and *stdio* in text mode.

`-Pidfile file`

Writes the PID *file* in the file.

`Single-step`

The instance is started in single step mode.

`-S`

Freezes the CPU during the start. The QEMU monitor can start with the command or *c cont* execution. With one instance can *stop* again pause.

Gdb-dev

Waits for a GDB connection on the device *dev* on. Typical compounds are based on TCP. Possible but also UDP, TCP and pseudo-stdio.

-S

1234th waiting on a GDB connection port This corresponds to the *option-gdb tcp:: 1234*.

-P *port*

for the port the GDB connection on *port* changes.

Item1-d ...

Writes log data into the log file / *tmp / qemu.log*. The possible *log-d* Items lists *on?*.

Hdachs-c, h, s [, t]

Forces for the virtual hard drive using the specified physical geometry (1 <= c <= 16383, 1 <= h <= 16, 1 <= s <= 63) and optionally causes the BIOS translation mode (t = *none*; *lba* or *auto*). With modern disks recognize QEMU and KVM chosen values. This option is useful for old DOS disks.

-L *path*

Defines the path to the directory containing the BIOS and VGA BIOS files.

BIOS file

Defines the filename of the BIOS image.

-Kernel-kqemu (QEMU to 0.11.x)

Unlocks the KQEMU virtualization. KQEMU is no longer supported QEMU 0.12.0.

-No-kqemu (QEMU to 0.10.6)

Disable the use of kqemu-kernel module. This is the case with the installation of Microsoft Windows as guest system is necessary. KQEMU is no longer supported QEMU 0.12.0.

-Enable-kqemu (QEMU to 0.11.x)

Enables the use of kqemu-kernel module. KQEMU is no longer supported QEMU 0.12.0.

-Enable-kvm

Enables the KVM hardware virtualization.

-No-kvm

Disables the KVM hardware virtualization. This is the case with the installation of Microsoft Windows as guest system is necessary.

-No-kvm-irqchip

Disables the KVM kernel mode PIC / IOAPIC / Lapic.

-No-kvm-pit

Disables the KVM kernel mode PIT.

-No-kvm-pit-reinjection

Disables the KVM kernel mode PIT interrupt reinjection

Pcidevice-host = bus: dev.func [, dma = none] [, name = string]

Put the specified PCI device to the host system the host system.

dma = none

There is no DMA Translations. By default *IOMMU* is implemented.

name = string

The parameter is the *string* used for logging output.

-Enable-nesting

If a virtualization solution within operated another virtualization solution, it is called *nesting*. This option allows the operation of a KVM KVM instance. For this, the host system must have an AMD processor. It is necessary to emulate the 64-bit CPU. Example: *qemu-system-x86_64-cpu Platte.img qemu64, + svm*. The *option-enable-nesting* was from QEMU 0.13.0 removed. Instead, *CPU, + svm* apply.

Nvram-FILE

Thus, the contents of a ia64-NVRAM is loaded from a file.

-Tdf

Setup a Time drift compensation for the guest.

-Kvm-shadow-memory MEGABYTE

The KVM MMU-shadowing is assigned to the given memory size.

-Path-mem FILE

Provides backing storage for the memory of the host system.

-Mem-prealloc

Of memory is allocated for the guest. This option is used together with *mempath*.

DOMID-xen-id

Sets the domain ID of the Xen host system. This option is only available when QEMU was configured with Xen support.

-Xen-create

Generates a domain using Xen Hyper calls and bypassing *xend*. This option should not be used with *xend* in combination. This option is only available when QEMU was configured with Xen support.

-Xen-attach

Adds the XEN virtual machine to an existing domain added. At the start of the instance is used *xend*. This option is only available when QEMU was configured with Xen support.

-No-reboot

Suppresses a reboot. Instead, the instance is terminated.

-No-shutdown

The authority will not end after the shutdown of the host system.

-Loadvm [tag | id]

Starts the instance directly to a previously saved state of the virtual machine. This corresponds *loadvm* in QEMU monitor.

-Daemonize

The instance is not so long separated from the Standardein-/ausgabe until the devices are of the instance in a position to accept connections. This option is useful for avoiding race conditions during initialization of the instance.

-Option-rom file

Loads the contents of the file specified as an optional ROM *file* for etherboot (see *option-boot n*).

-Clock method

Defines the method of the timer alarm. The available methods (*dynticks*, *hpet*, *rtc*, *unix*) *are-clock?* Displayed.

-Rtc [base = utc | local time | date] [, clock = host | vm] [, driftfix = none | slew]

Defines the settings of the system clock (Real Time Clock).

base = utc | date | local time

With the system clock is *utc* *base* = Coordinated Universal Time (Coordinated Universal Time) is set to. This is the default setting. For DOS and Windows guest systems is to specify *local time* *base* =. So Local Time will be set. The option *base* = *date*, the time of the system at the start of the value of *date*. Possible values are, for example 2008-04-17T16: 01:21 and 2008-04-17.

clock = host | vm

By default, the system of the instance to the system of the host system is provided. With the option *clock* = *vm* instance is running the system in isolation from the host.

driftfix = none | slew

This option is only *driftfix* x86 architectures. For problems with time-drift in Microsoft Windows with ACPI HAL option is to apply *driftfix* = *slew*. It is determined how many timer interrupts are not processed by the Windows guest system and attempts to apply it again.

-Local time (replaced by base-rtc = local time)

Sets the system clock to Local Time. This option allows DOS and Windows host systems, the correct time set. The default setting is UTC.

Start-date date (to be replaced by rtc-base = date)

This option sets the time of the system clock at the start of the value of *date*. Possible values are *now* (now), 2008-04-17T16: 01:21 and 2008-04-17. The default value is *now*.

Icount-[N | auto]

Enables the virtual instruction counter. The virtual CPU then executes an instruction every 2^N of ns. Is automatically set, so the CPU speed is automatically adjusted so that the virtual time of a few seconds in real time remains within.

Model-watchdog

Enables a virtual hardware watchdog device. With this device, the system is monitored. This must have the host system via an appropriate driver. Is activated in the host system, the Watchdog, an agent must make periodic requests to the hardware watchdog. If these questions from, so will the hardware watchdog device from a predetermined action. For example, while the guest system can be restarted (see-watchdog-action). The parameter *model* selects the model to be emulated. The models available lists

on-watchdog?. Currently, the models *i6300esb* (Intel 6300ESB) and *ib700* (IBAS 700) emulated. The choice depends on the support of the model by the host system.

Action-watchdog-reset | shutdown | poweroff | pause | debug | none

Defines the action when the watchdog timer alarm suggests. Possible actions are *reset* (default), *shutdown* (graceful shutdown), *poweroff*, *pause*, *debug* (output a debug message) or *none* (no action). The action is QEMU monitor with the command *watchdog_action* changed.

ECHR-numeric_ascii_value

Was given the *option Oceanographic*, is with the - the *ECHR* [Ctrl] + [A] change. The default setting is *0x01* and is the letter A. The other letters are coded according to their position in the alphabet. It is possible to provide this information in decimal or hexadecimal notation. If the key combination [Ctrl] + [T] be changed to the example, this is *ECHR-0x14 or 20 ECHR* stated.

C-virtioconsole

Sets the Virtio console. This option is *virt-device console* replaced and is provided for backward compatibility.

-Show-cursor

Display of the cursor.

-Tb-size n

Defines the TB size.

-Incoming tcp: ip: port

Enables the readiness for migration to be received over the network. The *ip* parameter specifies the IP address for reception of migration. If you specify 0, received on all interfaces. The *port* parameter defines the port.

-Nodefaults

Prevents the generation of default devices.

Chroot-dir

At the start of the instance will *chroot* to the specified directory with changed. This option is useful in combination with the *runas*.

Runas-user

At the start of the process instance to the specified user is transferred. This supports an increase in security.

Param-old-

For the ARM architecture mode with the old parameters will be activated.

Writeconfig-file

Saves the configuration *file* in the file.

Readconfig-file

Reads the configuration from the specified file.

-Nodefconfig

When starting QEMU normally be the settings of the configuration files *sysconfdir / qemu.conf* and *target*

`sysconfdir` / `ARCH.conf` used. The-suppressed `nodefconfig` this.

Options userspace emulation

For the user space emulation, the following programs: `qemu-i386` `qemu-x86_64`, `qemu-arm` `qemu-armeb`, `qemu-mips`, `qemu-mipsel`, `qemu-ppc` `qemu-ppc64`, `qemu-ppc64abi32`, `qemu-sparc`, `sparc32plus` `qemu`, `qemu-sparc64`, `m68k-qemu`, `qemu-cris`, `qemu-sh4`, `qemu` and `qemu-sh4eb-alpha`.

Syntax in Linux as the host system.

```
Host ~ $ qemu-i386 [-h] [-d] [-L path] [-s size] [-cpu model] \
[-G port] program [arguments ...]
```

Syntax for Mac OS X / Darwin as the host system.

```
Host ~ $ qemu-i386 [-h] [-d] [-L path] [-s size] \
program [arguments ...]
```

BSD syntax as the host system.

```
Host ~ $ qemu-sparc64 [-h] [-d] [-L path] [-s size] [bsd-type] \
program [arguments ...]
```

Standard Options

`-H`

Prints a help.

`-L path`

- Linux: Set the x86 ELF interpreter prefix (default = `/usr/local/qemu-i386`).
- Mac OS X / Darwin: Sets the library root path (default = `/`).
- BSD: Sets the library root path (default = `/`).

`-S size`

Defines the x86 stack size in bytes (Default = 524288).

`CPU model`

Selects the CPU `model`.`-Cpu?` Is list of possible models from a CPU.

`Bsd-type`

This option is only available on BSD and defined to emulate the BSD variants of `FreeBSD`, `NetBSD` and `OpenBSD` (default).

Debug options

`-D`

Enables logging (`log-Date` = `/tmp/qemu.log`).

`-P pagesize`

Simulates a host page size `page size` accordingly, in bytes.

`G-port`

Waits for connection of the debugger `gdb` for a port to `port`.

`Single-step`

The emulation is started in single step mode.

Strace-

Logs the system calls.

Environment variables

`QEMU_STRACE`

If this environment variable is assigned a value, the output system calls similar to the strace program.

Spice Options

Website: <http://www.spice-space.org>

Spice is a software and a protocol to support access to remote machines similar to VNC and RDP. Is being developed by the company Red Hat Spice, Inc. as open source. There is an implementation for QEMU / KVM.

`QXL-num [, ram = megs]`

QXL Turns one or more devices. *Num* with the number of devices is defined. Each device can be equipped with the *megs* of RAM are given in MB (default = 64 Mbyte). *The-QXL*, together with *the-spice* used.

```
-Spice [port =''port ='''][], sport sport''][], host =''host''] \
[, Ic = on | auto_glz | auto_lz | quic | GLZ | lz | off] \
[, Sv =''on | off''] \
[, Renderer =''oglpbuf oglpixmap + + cairo''] \
[, Play back-compression =''on | off''] \
[, Disable-ticketing] \
[, Password =''password''] \
[, Sslkey ='''key_file] \
[Sslcert ='''cert_file] \
[, Sslcafile ='''ca_file] \
[, Ssldhfile ='''dh_file] \
[, Sslpassword ='''pem_password] \
[, SSLCipherSuite ='''cipher_suite] \
[, Secure-channels = [all | ch0 + ch1 +...]] \
[, Unsecure-channels = [all | ch0 + ch1 +...]] \
[, Agent-mouse =''on | off'']
```

Enables the use of Spice. The *option-spice* is combined with *the-apply QXL*. Information on the current status Spice shows QEMU monitor the command *info spice.state*. Spice-statistics, the command displays *info on spice.stat*.

`[Port = port] [sport, sport =] [, host = host]`

With the option, the *host* network interface specified to the Spice listening on. If this option is not specified, Spice listens on all interfaces of the host system. With the options *port* or the port is given *sport*. For secure connections is to use *sport*. It is at least one of the two options specify *port* or *sport*.

`ic = on | auto_glz | auto_lz | quic | GLZ | lz | off`

The option is the *ic* Image Compression (Image Compression) configured. *Quic* SFALIC based on the algorithm. *Lz* enabled the Lempel-Ziv algorithm. This lossless compression method, sample image in GIF format used for. *GLZ* algorithm uses the Lempel-Ziv with a global dictionary. This dictionary is generated at run time. The options and *auto_glz* *auto_lz* *quic* is an automatic switching between *GLZ* or *lz* and accommodated. It is determined depending on the image properties, the best practice and applied. With the option *off* the image compression is disabled. The default setting is *auto_glz*. These settings can monitor the command *spice.set_image_compression* be changed in QEMU.

`sv = on | off`

The option `sv = on` (default value) enables the detection of video streams. It is then the lossy compression used. These settings can monitor the command `spice.set_streaming_video` be changed in QEMU.

`renderer = oglbuf oglpixmap + + cairo`

With this option selected, one or more renderers. To select `oglbuf` stand `oglpixma` and `cairo`. If more than one renderer given, specifies the order priority. The default value is `cairo`.

`playback-compression = on | off`

(De) Activates the playback compression. Here, the CELT algorithm (Constrained Energy Lapped Transform) is applied. CELT is a compression algorithm for audio data. The default value is `on`. In the QEMU monitor, the command `spice.set_playback_compression` this attitude changed later.

`disable-ticketing`

With this option, the ticketing is disabled. This client connections without a password is possible. This setting can monitor the command `spice.disable_ticketing` be changed in QEMU.

`password = password`

With the option of `password` content, the ticket is password for the client set. This password will not expire.

`sslkey = key_file`

This option is referred to the file containing the SSL private key.

`sslcert = cert_file`

This option is referred to the file containing the SSL certificate. This certificate may be a self-signed certificate or certificate chain and is not verified by the client. It is used only to send the public key from the server to the client.

`sslcafile = ca_file`

This option is made to the file with the CA (Certificate Authority) and the CRL (Certificate Revocation List).

`ssldhfile = dh_file`

This option is referred to the file with the symmetric Diffie-Hellman key.

`sslpassword = pem_password`

With the option `sslpassword` format is the passphrase to open the private key in PEM specified.

`SSLCipherSuite = cipher_suite`

With the option `SSLCipherSuite` the desired cipher suite is specified. A cipher suite defines the cryptographic protocol TLS (Transport Layer Security), which algorithms for establishing a data connection can be used. More information can be found at the URL <http://www.openssl.org/docs/apps/ciphers.html> .

`secure-channels = [all | ch0 + ch1 +...]`

`unsecure-channels = [all | ch0 + ch1 +...]`

Used by Spice transmission channels (channels) can be secured or unsecured. With the "secure-channels" are (`all`) or some channels encrypted all. With the `option-unsecure channels` are `all`) or certain channels not encrypted all (. There is the `main` channel, `display`, `input`, `cursor`, `play back` and

record. The default is supported for all channels of both the secured and the unsecured mode. The particular use of the *port* is used Spice options and *sport* from.

agent-mouse = on | off

The *option-mouse* defines *agent* as to whether the Spice Agent mode the client is to be used by the mouse. The default value is *on*. This setting is the command QEMUMonitor *spice.set_agent_mouse* changed.

Keyboard Shortcuts

In the window of an instance, the following keyboard shortcuts can be entered:

- [Ctrl] + [Alt] Release the mouse and keyboard.
- [Ctrl] + [Alt] + [1] Switch to console 1: Display of the host system.
- [Ctrl] + [Alt] + [2] Switch to console 2: QEMU monitor.
- [Ctrl] + [Alt] + [3] Switching to console 3: Serial output.
- [Ctrl] + [Alt] + [4] Switch to console 4: Parallel output.
- [Ctrl] + [Alt] + [H] Provides help in *the-Oceanographic*.
- [Ctrl] + [Alt] + [F] Toggle between full screen and windowed mode.
- [Ctrl] + [Alt] [U] Where the original window size restore.

In the virtual console of an instance are the shortcut keys [Ctrl] + [Cursor Up], [Ctrl] + [Cursor Down] to [Ctrl] + [Page Up] and [Ctrl] + [Page Down] Move in the text. The *option-ECHR* can be customized keyboard shortcuts.

When using the *option-key* combinations are as follows *Oceanographic* possible.

- [Ctrl] + [A] [H] Output of a help.
- [Ctrl] + [A], [?] Output of a help.
- [Ctrl] + [A], [X] Stops the instance.
- [Ctrl] + [A], [S] Saves changes to the virtual hard disk when you use *the-snapshot*.
- [Ctrl] + [A] [T] (De) Activates time marks in the console.
- [Ctrl] + [A] [B] Sends a break (*Magic Sysrq* in Linux).
- [Ctrl] + [A], [C] Switch between console and QEMU monitor.
- [Ctrl] + [A], [Ctrl] + [A] Sends the key combination [Ctrl] + [A].

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Startoptionen_von_QEMU_und_KVM "

This page has been accessed 17,458 times. This page was last updated on 26 January 2011 at 09:15 clock changed. Content is available under GNU Free Documentation License 1.2 .

QEMU QEMU Monitor KVM howto, manual, documentation readme

(Link to this page as [[QEMU-KVM-book / notes / QEMU Monitor]])

<<< | # # # | >>> | English

QEMU Monitor

The QEMU monitor supports manipulation of a running instance. With the key combination [Ctrl] + [Alt] + [2] leads to the QEMU monitor.

Standard Commands

```
(Qemu) help [cmd]
(Qemu) ? [Cmd]
```

Is a help for all commands or the specified command *cmd*.

```
(Qemu) commit device | all
```

Save the changes to the specified device or all devices. Prerequisite is the use of *option-snapshot*.

```
(Qemu) info
```

This command provides information about the state of the instance.

```
(Qemu) version info
```

Displays the version of QEMU.

```
(Qemu) info network
```

Displays information on VLANs and the associated devices.

```
(Qemu) info chardev
```

Displays information about the character devices.

```
(Qemu) info block
```

Displays information for the block devices (floppy disks, hard drives, CD / DVD) to.

```
(Qemu) info blockstats
```

Displays statistical information about the block devices.

```
(Qemu) info registers
```

Shows the CPU registers.

```
(Qemu) info cpus
```

Displays information for each CPU.

```
(Qemu) info history
```

Displays the command-line history.

```
(Qemu) info irq
```

Shows, if available, statistics for the interrupts to.

(Qemu) **info pic**

Displays the status of the i8259 chip (PIC) to.

(Qemu) **info usb**

Displays the emulated PCI devices.

(Qemu) **info tlb**

Shows the mapping of virtual to physical memory (only i386).

(Qemu) **info mem**

Displays the active virtual memory mapping (only i386).

(Qemu) **info hpet**

Shows if enabled the High Precision Event Timer (HPET) or not (only i386).

(Qemu) **info jit**

Displays dynamic compiler information.

(Qemu) **info kqemu** (QEMU to 00:11 .*)

Displays information about KQEMU.

(Qemu) **info kvm**

Displays information about the Kernel-based Virtual Machine.

(Qemu) **info numa**

Shows information about NUMA (Non-Uniform Memory Architecture) to.

(Qemu) **info usb**

Displays the virtual USB hub score attached USB devices.

(Qemu) **info USB host**

Show the USB devices on the host system.

(Qemu) **info profile**

Displays information about the internal profiler.

(Qemu) **info capture**

Displays information about active sound recordings (see *wavcapture*) to.

(Qemu) **info snapshots**

Displays a list of VM snapshots.

(Qemu) **info status**

Displays the state of the instance (running, pausing, Single Step Mode).

(Qemu) **info pcmcia**

Displays the status of the PCMCIA expansion (Personal Computer Memory Card International

Association) in the host system.

(Qemu) **info mice**

Indicates the mouse to the host system.

(Qemu) **info vnc**

Displays the state of the VNC server.

(Qemu) **info name**

Displays the name of the instance.

(Qemu) **info uuid**

Displays the UUID (Universally Unique Identifier) of the instance.

(Qemu) **info cpustat**

Displays statistical information about the CPUs.

(Qemu) **info Usenet**

Displays the status of the User-Network-stack connections (*see-net user, hostfwd = rule*).

(Qemu) **info slirp** (up QEMU 0.10.6)

Indicates if there is any SLIRP statistics.

(Qemu) **info migrate**

Displays the status of the migration.

(Qemu) **info balloon**

Displays information about ballooning.

(Qemu) **info qtree**

Displays the device tree.

(Qemu) **info qdm**

Displays the list of QDEV Device Models.

(Qemu) **info roms**

Displays information about the used ROM files (for example *vgabios-cirrus.bin* , " *pxe-e1000.bin*, " *bios.bin* ") to.

(Qemu) **q**

(Qemu) **quit**

Stops the instance.

(Qemu) **eject [-f] device**

Ejects removable media (CD / DVD, floppy) from. *The-f* option is pushing the ejection.

(Qemu) **change devicesettingdescription**

Changes the configuration of a device.

(Qemu) **change device filename [format]**

Swaps removable media (CD / DVD, floppy disk) from.

(Qemu) **change vnc display, options**

Changes the configuration of the VNC server. The syntax and *options* for *display* are the same as QEMU *vnc-start option*.

(Qemu) **change vnc password**

Changes the password of the VNC server. The password is significant up to 8 characters.

(Qemu) **Screendump file name**

Saves a screenshot as an image file in PPM format.

(Qemu) **logfile filename**

The expenditure will be written to the file *filename*.

(Qemu) **log item1 [, ...]**

Writes log data into the log file /tmp/qemu.log.

(Qemu) **savevm tag | id**

Creates a VM snapshot of the entire virtual machine. Condition, at least one image in qcow2 format. This virtual machine snapshot is a name (*tag*) or a progressive number (*id*) for each VM snapshot labeled. There is already a VM snapshot with the same *tag* or with the same *id*, this is overwritten. to the existing VM snapshot provides information on the QEMU monitor, the command *info snapshot s*. The *option-snapshot* may indeed VM snapshots are created, but these go to the end of the virtual machine lost after.

(Qemu) **loadvm tag | id**

Puts the entire virtual machine in the state of the previously saved VM snapshot. This corresponds to the start option *loadvm*.

(Qemu) **delvm tag | id**

Deletes a VM snapshot.

(Qemu) **single step [on | off]**

Turns the Single Step mode on or off.

(Qemu) **stop**

Can pause the instance (see-S).

(Qemu) **c**

(Qemu) **cont**

Lets *stop* the paused instance by continuing.

(Qemu) **gdbserver [device]**

Starts the GDB server on the specified device (default = *tcp: 1234*). Stopped, the GDB server *gdbserver* with *none*.

(Qemu) **x / fmt addr**

Creates a dump of the virtual memory address *addr*.

(Qemu) **xp / fmt addr**

Creates a dump of physical memory address *addr*. The *fmt* parameter defines how the data should be formatted. The syntax is */ {count} {format} {size}*

count

Defines the number of entries that are dump.

format

The format is defined as follows:

```
x (hexadecimal)
d (signed decimal)
u (decimal without sign)
o (octal)
c (character)
i (assembler instruction).
```

size

Size is defined as follows:

```
b (8 bit)
h (16 bit)
w (32 bit)
g (64 bit).
```

In the x86 CPU can *h* or *w* with the i-format can be defined to the size of the assembler instruction (16 or 32 bit) set. The following example creates a dump and places the result in the form of assembly language commands instructions of ten from the current execution position (the address in the EIP register) are:

(Qemu) **x/10i \$ eip**

The following example creates a hex dump of 80 16-bit values from the beginning of the video memory (text mode):

(Qemu) **xp/80hx 0xb8000**

```
(Qemu) p / fmt expr
(Qemu) print / fmt expr
```

Returns the value of the expression *expr*. It is used to format the result *fmt*.

(Qemu) **i fmt / addr**

Reads an I / O port.

(Qemu) **o / fmt addr value**

Returns the value of a *value* I / O port on.

(Qemu) **sendkey keys [hold_ms]**

The command codes *sendkey* sends the *keys-keys*. This can be sent via shortcuts that are intercepted by the host system. Several key codes for key combinations are associated with the minus sign. Example: *sendkey ctrl-alt-f1* The key codes of the letter keys are shown with the corresponding lower case letters. The key codes of the number keys are encrypted with the corresponding figures. Other key codes are: *shift*, *Shift_R* (right shift key), *old*, *Alt_R*, *ctrl*, *ctrl_r*, *menu* (Windows key), *esc*, *minus* (hyphen), *equal* (equal sign), *backspace*, *tab*, *ret* (return), *spc* (space) and *CAPS_LOCK*. The function keys are determined by the key codes *f1* to *f12*. The key codes are the numeric keypad *num_lock*, *SCROLL_LOCKS*, *kp_divide*

(divided by) *kp_multiply*, *kp_subtract*, *kp_add*, *kp_enter*, *kp_decimal*. The numbers on the numeric keypad are coded as follows: *kp_0*, *kp_1*, *kp_2*, *kp_3*, *kp_4*, *kp_5*, *kp_6*, *kp_7*, *kp_8* and *kp_9*. The cursor keys and the keys are also illustrated with the following key codes: *print*, *home*, *PgUp*, *PgDn*, *End*, *left*, *up*, *down*, *right*, *insert* and *delete*. The length of the key pressure is *hold_ms* given by (default 100 ms).

(Qemu) **system_reset**

Causes a system reset.

(Qemu) **system_powerdown**

Sends a system power-down event.

(Qemu) **sum addr size**

Calculates the checksum of a storage area.

(Qemu) **usb_add device**

Adds an additional USB device. Examples: *host usb_add*: and *usb_add bus.addr host: vendor_id: product_id*

(Qemu) **usb_del device**

Removes the USB device from the virtual device USB hub. *Bus.addr device* has the syntax. The QEMU monitor command shows *info usb* devices to the USB.

(Qemu) **device_add driver [, prop = value][,...]**

Adds a device added. The options are *the-device* option.

(Qemu) **device_del device**

Removes a device.

(Qemu) **MOUSE_MOVE dx dy [dz]**

Move the mouse cursor of the host system to the given coordinates *dx* and *dy*. Optional axle is rotated with the *scroll-dz*.

(Qemu) **mouse_button val**

Changes the state of a mouse button (1 = left, 2 = middle, 4 = right) in the host system.

(Qemu) **mouse_set index**

Change the active mouse device to the specified *index* number. Information on available mouse devices and their index numbers, the command *info mice*.

(Qemu) **wavcapture file [frequency [bits [channels]]]**

If the sound output to the host system and stores it in the file specified. More details define the frequency of the sample rate, sample bits and the number of channels (channels). The default values are sample-rate = 44100 Hz (CD quality), and number of bits = 16 Channels = 2 (Stereo).

(Qemu) **stop capture index**

Ends the recording of the given index. The index can be calculated with *capture info*.

(Qemu) **memsave addr size file**

Stores a dump of the virtual memory of size *size* from the address *addr* in *file* file.

```
(Qemu) pmemsave addr size file
```

Stores a dump of physical memory of size *size* from the address *addr* in *file* file.

```
(Qemu) boot_set boot device list (
```

Creates a new boot order set and thus overrides the values of *boot*. The values correspond to the parameters *of-boot*, but may vary according to machine type.

```
(Qemu) nmi cpu
```

Injected with a non-maskable interrupt (NMI) to the specified CPU.

```
(Qemu) migrate [-d] [-b] [-i] uri
```

Causes a migration to *uri*. The virtual machine is transmitted from a host system to another host system. If the *option-d* is not waiting for the integrity of the migration. The *-b* option enables the copying of the virtual disks to the destination host. This is useful if the image is not a common storage (shared storage) is located. The *option-i* is only a copy of the overlay files. This is useful if the same base images on source and destination host is available.

```
(Qemu) migrate_cancel
```

Cancels the current migration from.

```
(Qemu) migrate_set_speed value
```

Limits the maximum speed of migration in bytes.

```
(Qemu) migrate_set_downtime second
```

Defines the maximum tolerable downtime during migration in seconds.

```
(Qemu) drive_add [[<domain>:] <bus>:] <slot> \
    [<File = file> [, <if = type> [, <bus = n>] \
    [, <Unit = m> [, <media = d> [<index = i>] \
    [, <Cyls = c>, <heads = h>, <secs = s> [, <trans = t>]] \
    [<Snapshot = on | off> [, <cache = on | off>]
```

Adds a virtual drive to the PCI-PCI Storage Controller (see Adding an *option-drive*).

```
(Qemu) pci_add auto | [[<domain>:] <bus>:] <slot> nic | storage | host \
    [<Vlan = n> [, <macaddr = addr> [, <model = type>]] \
    [<File = file> [, <if = type> [, <bus = n>] ... \
    [<Host = 02:00.0> [, <name = string> [, <dma = none>]
```

The command *pci_add* devices allows you to add hot-started the host system (Linux). This requires the host system, the kernel modules are loaded and *pci_hotplug acpiphp*. Examples:

```
(Qemu) pci_add 0 model = e1000 nic
```

Adds a virtual network card.

```
(Qemu) pci_add 0 storage file = Platte.img, if = scsi
```

Adds a virtual hard disk.

```
(Qemu) pci_del [[<domain>:] <bus>:] <slot>
```

Removes the specified PCI device on the fly.

```
(Qemu) host_net_add tap | user | socket | vde | dump [options]
```

Adds a VLAN client added (see *option-net user*).

```
(Qemu) host_net_remove vlan_id name
```

Removes a VLAN client.

```
(Qemu) host_net_redir (replaced by hostfwd_add)
```

Redirects TCP or UDP connections from host to host system to. This *user-net* option is required.

```
(Qemu) netdev_add [user | tap | socket], id = str [, prop = value ][,...]
```

Adds a network device added. The options are the same as with *option-netdev*.

```
(Qemu) netdev_del id
```

Deletes a network device.

```
(Qemu) hostfwd_add [vlan_id name] \
[Tcp | udp]: [hostaddr]: host port [guestaddr]: guest-port
```

Initiates a TCP or UDP connection from host to host system further. Requirement for that is *the-net user*. In addition to the parameters *of-net user*, *hostfw*, the VLAN ID can be specified. This is more virtual network cards required at (see-*net nic*, *vlan = n*).

```
(Qemu) hostfwd_remove [vlan_id name] [tcp | udp]: [hostaddr]: host port
```

Removes a port redirect (see-*net user*, *hostfwd = rule*).

```
(Qemu) balloon value
```

Calls to change the virtual machine on which the storage allocation (in MB).

```
(Qemu) set_link name [on | off]
```

Connects the virtual network and the network adapter *name*.

```
(Qemu) watchdog_action [reset | shutdown | poweroff | pause | debug | none]
```

Changes the watchdog action (see *option-watchdog-action*).

```
(Qemu) acl_show aclname
```

Was QEMU / KVM with support for SASL and ACL compiled, this command lists all the appropriate rules of the ACL (Access Control List) and the default policy. There are two named ACLs: *vnc.x509dname* the Distinguished Name During tests, checked *vnc.username* to the SASL user name (see-*vnc*).

```
(Qemu) acl_policy aclname allow | deny
```

Was QEMU / KVM with support for SASL and ACL compiled, this command sets the default policy of access control list. This uses default policy if no access to the defined rules. The default policy is always at the start set to *deny*.

```
(Qemu) acl_add aclname match allow | deny [index]
```

Was QEMU / KVM with support for SASL and ACL compiled, this command adds a match rule to the ACL. This allows or denies access. This requires either the user or the *x509 distinguished name* match. Optional wildcards are possible. For example, * @ EXAMPLE.COM allows all users of the Kerberos realm EXAMPLE.COM access. Normally the rule is appended to the end of the access control list. the *index* parameter with the rule can be inserted a specific point on.

```
(Qemu) acl_remove aclname match
```

Was QEMU / KVM compiled with support for SASL and ACL, this command removes a rule from the Access

Control List.

(Qemu) **acl_reset aclname**

with QEMU / KVM support for SASL and ACL was compiled, this command will remove all the matching rules from the access control list and sets the default policy back to *deny* again.

(Qemu) **mce cpu bank status mcgstatus addr misc**

Injects a Machine Check Exception (MCE). This is possible only on x86-based CPUs. A machine check exception is a report of recent CPU (s) from Intel and AMD hardware problems.

cpu

The MCE is injected for the specified CPU. If only one CPU is emulated, *0* specify here. When multiple CPUs (see *option-smp*) Counting starts with *0*.

bank

Each CPU is assigned to four or five banks. A bank is a group of error registers. This error registers can each be activated or deactivated. Normally all error register of all banks are activated. The number and content of the benches depends on the CPU. Each bank has four registers. In an AMD CPU, the banks have the following meanings:

- Bank 0: Date cache
- Bank 1: Instruction cache
- Bank 2: Bus Unit,
- Bank 3: Load Store Unit
- Bank 4 North Bridge and DRAM.

status

The bank status register is 64 bits long.

mcgstatus

The global status register contains information whether a MCE was reported.

- Bit 0: RIPV - Restart IP valid flag,
- Bit 1: EIPV - Error IP valid flag,
- Bit 2: MCIP - Machine check in progress,
- Bit 3 to 63: Reserved.

addr

Sets the memory address of the exception is defined.

misc

This option is for additional description of the MCE.

(Qemu) **getfd fdname**

Receives a file descriptor via Unix sockets for *SCM_RIGHTS* and assigns it a name. QEMU monitor commands use this name. Such a file descriptor is for adding and removing network cards during the current host system (hotplug) are required.

(Qemu) **closefd fdname**

Includes a *SCM_RIGHTS* received via file descriptor.

(Qemu) **block_passwd device password**

Sets the password for the encrypted device *device*.

(Qemu) **qmp_capabilities**

QMP-enabled capabilities. QMP (QEMU Machine Protocol) is an on JASON (http://www.json.org based management interface that lets applications communicate with the QEMU instance) allows (see http://www.linux-kvm.org/page / monitor protocol).

(Qemu) **cpu index**

Sets the default CPU.

(Qemu) **cpu_set cpu [online | offline]**

Changes the status of the specified CPU.

The QEMU monitor understands integer expressions for any integer argument. Register name preceded by \$ (dollar sign) contain the values of CPU registers.

Spice commands

Website: http://www.spice-space.org

Spice is a software and a protocol to support access to remote machines. Is being developed by the company Red Hat Spice, Inc. as open source. There is an implementation for QEMU / KVM.

(Qemu) **info spice.state**

Information on the current status Spice. These include the addresses of connected clients, Tickening, image compression, video streaming and playback compression.

(Qemu) **info spice.ticket**

The command *info spice.ticket* informed of possible end times of the tickets (see *spice.set_ticket*).

(Qemu) **info spice.stat**

Spice Displays statistics.

(Qemu) **info spice.rtt_client**

Displays statistics (minimum, maximum and average) of client response times (Round Trip Time) (see command *spice.ping_client*).

(Qemu) **spice.set_image_compression on | auto_glz | auto_lz | quic | GLZ | lz | off**

With the command *spice.set_image_compression*, the Image Compression (Image Compression configured. The options are the *spice of-ic*.

(Qemu) **spice.set_streaming_video on | off**

The command *spice.set_streaming_video* (de-) activates the detection of video streams and the resulting lossy compression (see *Option-spice sv*).

(Qemu) **spice.set_playback_compression on | off**

(De) Activates the playback compression (see *Option-spice-compression playback*).

(Qemu) **spice.set_ticket password \ [Expiration = seconds [, connected = keep | disconnect | fail]]**

The command *spice.set_ticket* is the ticket (one-time password). A blank password prevents any connection. Optional can be defined with the *expiration* of the password expiration time. A value of 0 is the infinite expiration time. Furthermore, it can influence the behavior at the connection set up. = *Keep connected* with the password, the connection of the end of the hold. With *connected = disconnect* the connection is terminated. The command *info spice.ticket* informed about the current expiry times.

```
(Qemu) spice.set_ticket64
```

The command corresponds to the command *spice.set_ticket64* *spice.set_ticket*. However, for the ASCII password using *base64*.

```
(Qemu) spice.disable_ticketing
```

This command connection is possible without a client password (see-*spice disable-ticketing*).

```
(Qemu) spice.set_agent_mouse on | off
```

The command *spice.set_agent_mouse* defines whether the agent of Spice-mouse mode should be used by the client (see *option-spice agent-mouse*).

```
(Qemu) spice.reset_stat
```

Resets the Spice Statistics.

```
(Qemu) spice.ping_client [on [interval] | off]
```

To check the response time of the client, use the command *spice.ping_client*. If only one ping allowed, no options are necessary. If the ping to be repeated), the option is *on* and the interval (default = 1 second indicated. Finished *off* with the ping. The relevant statistics, the command *info spice.rtt_client*.

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_QEMU-Monitor "

This page has been accessed 14,293 times. This page was last updated on 17 January 2011 at 18:51 clock changed. Content is available under GNU Free Documentation License 1.2 .

qemu-img convert kvm-img create commit info qcow2 sparse vmdk file, readme manual howto help documentation

(Link to this page as [[QEMU-KVM-book / notes / qemu-img]])

<<< | # # # | >>> | English

Contents

1	qemu-img
01.01	check
2.1	create
1.3	convert
4.1	commit
1.5	snapshot
1.6	info
1.7	rebase
1.8	resize
2	Supported image formats
2.1	raw
2.2	host_device
2.3	qcow2
2.4	qcow
2.5	cow
6.2	vmdk
7.2	vdi
8.2	cloop
09.02	nbd
2:10	parallels
2:11	vvfat
2:12	vpc
2:13	bochs
2:14	dmg

qemu-img

qemu-img (kvm-img) is a tool to generate and convert image files.

```
Host ~ $ qemu-img command [options]
      -H
```

Displays a help.

For *command*, the following commands with the appropriate options are used.

check

```
Host ~ $ qemu-img check [-f fmt] filename
```

Verify an image file (*file name*). Optionally, the *format-f* are specified. Currently only the format qcow2 is supported. Example:

```
Host ~ $ qemu-img check file.img
No errors were found on the image.
```

create

```
Host ~ $ qemu-img create [-e] [-6] [-F base_fmt] [-b base_image] \
      [-F fmt] [-o option] filename [size]
```

Generates an image with the file name *file name* and size of the *size*.

filename

Name of the generated image file.

size

Defines the size of the image file in kilobytes. Optionally, the suffix *M* (megabyte = 1024 * 1024), *Gigabyte* = 1024 * 1024 * 1024 or *T* (terabyte = 1024 * 1024 * 1024 * 1024) are indicated (*G*). When the size of the image file specified with *the-o size* is obsolete with the *size*.

-F *fmt*

Defines the format of the image file.

-E

Encrypted image. This is only qcow and qcow2 format possible. The encryption is done with AES (128 bit key), making it very safe. For maximum security, a password must be used with 16 characters.

-6

Forces at the vmdk format compatibility level 6 for the destination image file.

-B *base_image*

If *base_image* an existing image file specified with, then stores the generated image, only the differences to these existing image file. The generated image is called an overlay file. One size is not for the overlay file is required. The base image (*base_image*) is not changed unless there is entered *commit*.

F-base_fmt

Specifies the format of the base file. Normally, this option is not necessary because *qemu-img* format normally recognizes that.

-O *options*

Allows you to specify additional options. The format is *name = value*. Multiple options are separated by commas. *With-o?* Get a list of supported switches.

```
Host ~ $ qemu-img create-o?
```

Supported options:
size virtual disk size

Example:

```
Host ~ $ qemu-img create-f-o size = 1G qcow2 file.img
```

00:13 From QEMU following syntax is supported:

```
Host ~ $ qemu-img create [-f fmt] [-o option] filename [size]
```

The existing *options-e*, *-6*, *-F* and *-b* are as relevant information behind the *option-o*. These values depend on the type of image formats. In this example, an overlay file is created.

```
Host ~ $ qemu-img create-f-o qcow2 backing_file = Base Platte.img Platte.img 1G
```

convert

```
Host ~ $ convert [-c] [-e] [-6] [-f fmt] [-O output_fmt] [-o option] \'''[-B output_base_image] [filename2 [...]] output_filename file name'''
```

Copies an image in a new image with a different format.

filename

Name of the source file.

output_filename

Name of the generated image file.

-F fmt

Specifies the format of the source file. Usually recognizes *qemu-img* format. Therefore, this information is often not necessary.

-O output_fmt

Defines the target format.

B output_base_image

It is possible for a base image with the same content in the Image Copy *output_base_image* to. The path, image format and other original can be changed in *base_image* to the file.

-C

Allows compression when qcow and qcow2 format. If, after compressing sectors rewritten, they are not compressed.

-E

Enables encryption when qcow and qcow2 format. The encryption is done with AES (128 bit key), making it very safe. For maximum security, a password must be used with 16 characters.

-6

Forces at the vmdk format compatibility level 6 for the destination image file.

-O options

Allows you to specify additional options. The format is *name = value*. Multiple options are separated by commas. *With-o?* Get a list of supported switches.

```
Host ~ $ qemu-img convert-o? File.img KonvertierteDatei.img
Supported options:
size virtual disk size
```

Example:

```
Host ~ $ qemu-img convert-O-o size = 1G qcow file.img \
KonvertierteDatei.img
```

00:13 From QEMU following syntax is supported:

```
Host ~ $ qemu-img convert [-c] [-f fmt] [-O output_fmt] [-o options] \
filename [filename2 [...]] output_filename'''
```

The existing *options-e*, *-6* and *-B* as equivalent data behind the *option-o*. The details depend on the used image formats (see below).

commit

```
Host ~ $ commit [-f fmt] filename
```

Saves changes in the base image.

-F fmt

Specifies the format of the image.

filename

Name the image file.

snapshot

```
d snapshot] filename ~-l |-a snapshot | snapshot c | $ snapshot [-host
```

Managed VM snapshots.

snapshot

Is the name of VM snapshots.

-A snapshot

The state of the specified VM snapshot *snapshot* restore.

C-snapshot

Creating a VM snapshot with the *snapshot* name.

Snapshot-d

Deleting the VM snapshot with the *snapshot* name.

-L

List all VM snapshots of an image.

filename

Name the image file.

info

Host ~ \$ **info** [-f fmt] filename

Display information (size, VM snapshots, ...) to an image.

-F fmt

Specifies the format of the image.

filename

Name the image file.

rebase

Host ~ \$ **qemu-img rebase** [-f fmt] [-u] -b backing_file [-F backing_fmt] filename

00:13 From QEMU command the *rebase* supported. Thus, the overlay file a new base image will be assigned. By default, there must be the current and new base image. The new base images is compared with the old base image. If differences are found, the data from the old base image to the new base image is transmitted.

-F fmt

Specifies the format of the old base image. Usually recognizes *qemu-img* format. Therefore, this information is often not necessary.

-U

Enables an insecure rebasing. It is assumed that the old and the new base image match exactly. This is useful for renaming or relocating the base images.

-B backing_file

Sets the new base image is specified.

-F-base_fmt

Specifies the format of the base file. Normally, this option is not necessary because *qemu-img* format normally recognizes that.

filename

By *filename* is the overlay indicated that the new base image is to be assigned.

resize

Host ~ \$ **qemu-img resize** file name [+ | -] size

00:13 From QEMU command will *resize* support. With *resize* format, the capacity of a virtual disk in raw changed. Before you shrink an image that must be shared in the virtual machine a correspondingly large area at the end of the virtual hard disk. This would cause data loss. If an image enlarged to the extent the host system, this new storage area to partition and format.

filename

With the *filename* image is shown to scale.

size

Defines the new size or the change in size in kilobytes. Optionally, the suffix M (megabyte = 1024 * 1024), are given G (gigabyte = 1024 * 1024 * 1024) or T (terabyte = 1024 * 1024 * 1024 * 1024).

Supported image formats**Raw**

The raw format is the default format. It is simple and can easily convert. a new, blank image in raw format to a certain size, for example, produced 10 GB, it occupies in older file systems exactly this size (10 GB). Newer file systems support more effective management of files, containing the uncovered blocks. Such files occupy in these file systems such as just the space they occupy on blocks. A file of unused blocks is **sparse** file. Images in *raw* format files can be created as sparse.

host_device

For block or other devices, the effective management of unused blocks in files do not support, the format *host_device* apply.

qcow2

The format is the most versatile *qcow2* format and replaces the old format from *qcow*. Images in *qcow2* format are dynamic. Their size depends on their level. When *qcow2* format file size is independent of whether the file system for efficient management of sparse files supported. *Qcow2* supports multiple storage of system states of the virtual machine (VM snapshot). Furthermore, encryption (AES) and compression (zlib) is possible. In *qcow2* content is stored in the clusters. Each cluster contains a number of 512 byte sectors. There are options of *qemu-img* (-o) supports the following:

backing_file = filename

Filename of the base image (*see-img create qemu*).

backing_fmt = fmt

Image format of the base image (*see-img create qemu*).

encryption = on | off

Encrypted image. The encryption is done with AES (128 bit key), making it very safe. For maximum security, a password must be used with 16 characters.

`cluster_size = alue`

Changes the default cluster size (512 bytes). There are values between 512 and 2MByte possible. Smaller values optimize the file size. Larger values result in better performance.

`preallocation = off | metadata`

An image with assigned meta-data is initially larger, but has a better performance when zooming.

qcow

qcow is the old QEMU image format. Images in *qcow* format are dynamic. Their sizes depend on their level. In *qcow* the file size does not depend on whether the file system for efficient management of sparse files allowed. Still supported *qcow* encryption and compression. There are options of *qemu-img (-o)* supports the following:

`backing_file`

See *qcow2*.

`encryption`

See *qcow2*.

cow

The old format copy-on-write is available only because of compatibility with older versions of QEMU. It does not work under Microsoft Windows.

vmdk

vmdk is the standard format of VMware Workstation. There are options of *qemu-img (-o)* supports the following:

`backing_fmt`

See *qcow2*.

`compat6`

Forces at the *vmdk* format compatibility level 6 for the destination image file.

vdi

vdi is the standard format of VirtualBox.

cloop

The format *Compressed loop* is compressed CD-ROM images used, for example Knoppix-CD/DVD-ROMs.

nbd

nbd), the format of the NBD protocol (Network Block Device.

Parallels

Parallels is the standard format of virtualization solutions from Parallels, Inc.

vvfat

With *vvfat* disks are virtual FAT meant.

vpc

VPC is the standard format for images of Microsoft Virtual PC.

bochs

bochs is the format of the free and AMD64 x86 emulator Bochs.

dmg

dmg is a disk image file format under Mac OS X and is used mostly for distribution of software over the Internet. These images can be uncompressed or compressed. Uncompressed *dmg* files can be read directly from *qemu img*. Compressed *dmg* files can be unpacked *dmg2img* the tool with.

<<< | # # | >>> http://qemu-buch.de

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_qemu-img "

This page has been accessed 14,252 times. This page was last updated on 22 January 2011 at 13:51 clock changed. Content is available under GNU Free Documentation License 1.2 .

qemu-io, kvm-io - QEMU disk exerciser, manual commands help howto

(Link to this page as [[QEMU-KVM-book / notes / qemu-io]])

<<< | # # # | >>> | English

Contents

1 qemu-io
1.1 Option
2.1 Commands
1.2.1 aio_flush
1.2.2 aio_read
1.2.3 aio_write
1.2.4 alloc
1.2.5 close
1.2.6 flush
1.2.7 help
1.2.8 info
1.2.9 length
1.2.10 multi-write
1.2.11 Open
1.2.12 quit
1.2.13 read
1.2.14 readv
1.2.15 truncate
1.2.16 write
1.2.17 writev

qemu-io

qemu-io (kvm-io) is a diagnostic and manipulation program (virtual) memory media. *qemu-io* is still at a very early stage of development. This tool is called with *qemu-io* and options.

```
Host ~ $ qemu-io [-h] [-V] [-Crsnm] [-c cmd] ... [File]
```

Options

-C, - create (up QEMU 0.12)

Creates a new disk image, should the specified file does not exist.

-C, - cmd

Runs the specified command *qemu-io-shell*.

-R, - read-only

Export the image or read-only device.

-S, - snapshot

Uses a snapshot file.

-N, - nocache

Disable the cache of the host system.

-G, - growable

Allows the file may be increased (only for the minutes).

-M, - misalign

Misalignment in allocations for *O_DIRECT*.

-K, - native-aio

Uses the kernel AIO implementation for Linux as the host system.

-H, - help

It displays a help message and then exits.

-V, - version

It displays the version and then exits.

Commands

With no options specified, you get to command mode.

```
Host ~ $ qemu-io
qemu-io>
```

aio_flush

Completes all outstanding AIO requests.

```
qemu-io> aio_flush
```

aio_read

Asynchronous reading a predetermined number of bytes.

```
qemu-io> aio_read [-CQV] [-P pattern] off Len [len ..]
```

-C

Is reporting statistics in machine-readable form.

-P

Used a data sample to check the read data.

-V

Outputs the contents of the buffer to standard output.

-Q

Not seeing any I / O statistics (Quite mode).

Example:

```
qemu-io> aio_read-v 512 1k 1k
```

Reads two KByte from the offset of 512 and outputs the buffer to standard output.

aio_write

In the opened file from the specified offset a pattern (default = *0xcdcdcdcd*) in the specified length in bytes written asynchronously. Then the command is applied *aio_flush*.

qemu-io> **aio_write** [CQ] [-P pattern] off Len [len ..]

-P

There are different patterns used to fill the file.

-C

Is reporting statistics in machine-readable form.

-Q

Not seeing any I / O statistics (Quite mode).

Example:

qemu-io> **aio_write** 512 1k 1k

Writes two KByte from the offset of 512 bytes in the open file.

alloc

Tests if a given sector is present in the file.

qemu-io> **alloc** off [sectors]

close

Closes the currently opened file.

qemu io-> **close**

flush

Writes all the in-core file states on the hard disk.

qemu io-> **flush**

help

Display a help for a particular or for all commands.

qemu-io> **help** [command]

info

View information about the current image file.

qemu-io> **info**

length

Returns the length of the current image file.

qemu-io> **length**

multi write

Execute multiple write requests to the open file with a command through. For each write request, to the number of bytes, the offset and the buffer is predetermined. This) is a byte-pattern (default = 0xcdcdcdcd used.

qemu-io> **multi write** [CQ] [-P pattern] off Len Len ..] [[; off Len [len ..]..]

-P

There are different patterns used to fill the file.

-C

Is reporting statistics in machine-readable form.

-Q

Not seeing any I / O statistics (Quite mode).

Example:

```
qemu-io> MultiWriter 512 1k 1k, 4k 1k
```

In this example, 2 kbytes from an offset of 512 bytes and 1 kbyte from an offset of 4 Kbytes written to the open file.

Open

Opens an image file to commands from *qemu-io* to work with. Here, the path specified.

```
qemu-io> open [-pretest] [path]
```

-C

Creates a new file when the specified file does not exist.

-R

Open the file read-only.

-S

Uses a snapshot file.

-N

Disable the cache of the host system.

-G

Allows increasing the file (only for protocols).

Example:

```
qemu-io> open-Cn / tmp / data
```

Creates and opens a file read-only. The host cache is deactivated.

quit

Exit the program.

```
qemu-io> quit
```

read

Reads a number of bytes from the specified offset in a buffer.

```
qemu-io::: read [-abCpqv] [-P pattern [s-off] [-l len]] Len off
```

-B

First reads the data from the VM state instead of the virtual hard disk.

-C

Is reporting statistics in machine-readable form.

-L

Length of the pattern for data validation (only *with-P*).

-P

Bdrv_pread used to read the file.

-P

Used a data sample to check the read data.

-Q

Not seeing any I / O statistics (Quite mode).

-S

Starts at the specified offset to the data verification (only *with-P*).

-V

Outputs the contents of the buffer to standard output.

Example:

```
qemu-io: read-v 512 1k
```

Returns the file a range of one Kbytes from the offset of 512 bytes from the standard output.

readv

Reads a number of bytes from the specified offset. If more than one byte-range is defined, is written in several buffers.

```
qemu-io> readv [-CQV] [-P pattern] off Len [len ..]
```

-C

Is reporting statistics in machine-readable form.

-P

Used a data sample to check the read data.

-V

Outputs the contents of the buffer to standard output.

-Q

Not seeing any I / O statistics (Quite mode).

Example:

```
qemu-io> readv-v 512 1k 1k
```

Returns the file from two areas of one Kbytes from the offset of 512 bytes to the standard output.

truncate

Cuts the current file at the specified offset.

```
qemu-io> truncate off
```

write

Writes a number of bytes from the specified offset from a buffer. This) is a byte-pattern (default = *0xcdcdcdcd* used.

```
qemu-io> write [-abCpq] [-P pattern] Len off
```

-B

Writes first place the data in the VM status on the virtual hard disk.

-P

Bdrv_pwrite used to write to the file.

-P

There are different patterns used to fill the file.

-C

Is reporting statistics in machine-readable form.

-Q

Not seeing any I / O statistics (Quite mode).

Example:

```
qemu-io> write 512 1k
```

Writes a KB from the offset of 512 bytes in the open file.

writev

Writes to the file opened a number of bytes from the specified offset of several buffers. This) is a byte-pattern (default = *0xcdcdcdcd* used.

```
qemu-io> writev [CQ] [-P pattern] off Len [len ..]
```

-P

There are different patterns used to fill the file.

-C

Is reporting statistics in machine-readable form.

-Q

Not seeing any I / O statistics (Quite mode).

Example:

```
qemu-io> write 512 1k 1k
```

It will be written two KByte from the offset of 512 bytes in the open file.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_qemu-io "

This page has been accessed 3445 times. This page was last updated on 19 January 2011 at 17:01 clock changed. Content is available under GNU Free Documentation License 1.2 .

nbd qemu and kvm-nbd are tools to export of image files using the NBD protocol (Network Block Device).

(Link to this page as [[QEMU-KVM-book / notes / qemu-nbd]])

<<< | # # # | >>> | English

qemu-nbd

nbd qemu and *kvm-nbd* are tools to export of image files using the NDB Protocol (Network Block Device). NDB is a simple protocol for Linux to export a block devices over the network. *Qemu-nbd* is in the package *qemu* and *kvm kvm-nbd* package included. Both have the same functions.

Host ~ \$ **qemu-nbd [OPTION] ... file**

file is the name of the image file.

-P, - port = port

Defines the port for the NBD server (default = 1024).

-O, - offset = offset

Defines the offset in the image.

-B, - bind = iface

Connections are only allowed on the specified interface (default = 0.0.0.0).

-K, - socket = path

The connection is made possible through a UNIX domain socket. The *path* parameter defines the path to the socket.

-R, - read-only

The image is exported read-only.

-P, - partition = num

It is the partition with the specified number *num* exports only.

-S, - snapshot

Changes are not written to the image, but in temporary files.

-N, - nocache

The host cache is disabled.

-C, - connect = dev

Connects the image with the device *dev*.

-D, - disconnect

Disconnects the specified device.

-E, - shared = num

The device can be used simultaneously by multiple clients. The maximum number defines the value of

num (Default = 1).

-T, - persistent

The program will not terminate when the last connection.

-V, - verbose

It displays additional debugging information.

-H, - help

Display of support.

-V, - version

Display version.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_qemu-nbd "

This page has been accessed 3969 times. This page was last updated on 27 September 2010 at 06:34
clock changed. Content is available under GNU Free Documentation License 1.2 .

libvirt 0.8.2, virsh, virt-manager, virt-install, virt-clone man page, cloud computing

(Link to this page as [[QEMU-KVM-book / notes / libvirt]])

<<< | # # # | >>> | English

Contents <ul style="list-style-type: none"> 1 libvirt 0.8.2 <ul style="list-style-type: none"> 1.1 virsh <ul style="list-style-type: none"> 1.1.1 Name 1.1.2 Synopsis 1.1.3 Description 1.1.4 Notes 1.1.5 General commands 1.1.6 Commands to Manage Domains 1.1.7 Commands to Manage Devices 1.1.8 Commands to manage Host Devices 1.1.9 Commands for managing storage pools 1.1.10 Commands for managing virtual disks 1.1.11 Commands for managing virtual networks 1.1.12 Commands for security 1.1.13 Environment Variables See also 1.1.14 1.1.15 authors COPYRIGHT 1.1.16 1.1.17 LICENSE 1.1.18 Bugs Referenced by 1.1.19 2.1 virt-install <ul style="list-style-type: none"> 1.2.1 Name 1.2.2 Synopsis 1.2.3 Description 1.2.4 Options <ul style="list-style-type: none"> 1.2.4.1 General 1.2.4.2 Full virtualization 1.2.4.3 types of virtualization 1.2.4.4 Installation methods 1.2.4.5 Storage configurations 1.2.4.6 network configurations 1.2.4.7 Configurations for graphics output 1.2.4.8 Other Options 1.2.5 Examples 1.2.6 Authors 1.2.7 Bugs 1.2.8 Copyright 1.2.9 See also Referenced by 1.2.10 1.3 virt-clone <ul style="list-style-type: none"> 1.3.1 Name 1.3.2 Synopsis 1.3.3 Description 1.3.4 Options <ul style="list-style-type: none"> 1.3.4.1 General 1.3.4.2 Storage configurations 1.3.4.3 network configurations 1.3.4.4 Other Options 1.3.5 Examples 1.3.6 Authors 1.3.7 Bugs 1.3.8 Copyright 1.3.9 See also 4.1 virt-viewer <ul style="list-style-type: none"> 1.4.1 Name 1.4.2 Synopsis 1.4.3 Description 1.4.4 Options 1.4.5 Examples 1.4.6 Authors 1.4.7 Bugs 1.4.8 Copyright 1.4.9 See also 1.5-manager virt <ul style="list-style-type: none"> 1.5.1 Name 1.5.2 Synopsis 1.5.3 Description 1.5.4 Options 1.5.5 Authors 1.5.6 Bugs 1.5.7 Copyright 1.5.8 See also 1.5.9 Referenced by

libvirt 0.8.2

Website: <http://libvirt.org>

Libvirt library to be the case of host-guest systems as nodes and systems referred to as domains.

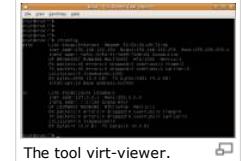
virsh

Name

virsh - User interface for managing

Synopsis

```
~ # Virsh subcommand [args]
```



The tool virt-viewer.



The Virtual Machine Manager.

Description

The program is *virsh* the general user interface for managing guest domains. The program allows you to create, pause, and shutdown domains. Furthermore, can be listed current domains. The library is one of the *libvirt* language written in C Toolkit. It supports interaction between virtualization of current versions of Linux and other operating systems. The Open Source Software *libvirt* under the *GNU Lesser General Public License*. Virtualization of Linux operating systems is the ability to operate simultaneously to multiple instances of an operating system on hardware, where the resources are controlled by a Linux instance. The aim of the library *libvirt* is to develop a standard API for Xen, QEMU, KVM, VMware ESX, VMware Server, VirtualBox or other virtualization solutions. The general syntax is:

```
~ # Virsh command domain-id [OPTIONS]
```

For *command* is one of the commands listed below use. With *domain-id* domain is the internal numeric ID or name of the intended. Depending on the *OPTIONS* command with certain options are specified. There are some exceptions to the syntax for specific commands. These are the commands that manage all the domains that control the host machine, relating to the hypervisor. The program can accurately *virsh command* command on the command line to run to a pass with. If no command is specified, an interpreter is started. In the shell of this interpreter *virsh* commands are entered. Exits the command interpreter is to *quit*.

Notes

All *virsh* operations based on the *libvirt* library. This library extends the *virsh* commands to *xend*, *QEMU* and other virtualization solutions *libvirt* support further with. It should start to *libvird*. This is usually done by typing / *etc* / *init.d* / *libvirt-bin start*.

Most commands require *virsh* root privileges to communicate with the hypervisor. Continue to work asynchronously most *virsh* commands. That is, the command interpreter, the shell of the free again, is the associated action is not necessarily finished. Considerations include execution times of at least 30 seconds at many operations with domains, such as the establishment and shutdown. The *virsh-list* command shows the status of the command processing. By reopening the display is updated.

General orders

The following commands are not applicable to a specific domain.

```
virsh # help [command]
```

The *help* command gives brief information about all commands in a. With the *help command* command is a detailed help on the specified output *command*.

```
virsh # quit
```

Ends the interactive terminal.

```
virsh # version
```

Provides information on the versions used during the build. Example:

```
~ # Virsh version
Connecting to uri: qemu: // / session
Compiled against the library: libvirt 0.8.2
Use Library: libvir 0.8.2
Use API: QEMU 0.8.2
Running hypervisor: QEMU 0.13.0
```

```
virsh # cd [directory]
```

Changes the path of the current directory. The default directory is the home directory. If the variable *HOME* is not defined, the default directory is the main directory. This command is available only in interactive mode.

```
virsh # pwd
```

Displays the path of the current directory.

```
virsh # connect URI [- readonly]
```

Connects to the specified hypervisor (again) ago. The option - *read only* allows read-only connection. About the URI parameter in this internal command, the connection is set to hypervisors. The documentation at the URL <http://libvirt.org/uri.html> and <http://libvirt.org/drivers.html> listen to the supported values. Often following URIs are used:

```
test: // / default
```

This URI connects you with the local dummy hypervisor (<http://libvirt.org/drvtst.html>). This simulates a running domain. It is recommended that the first *virsh* commands to try out this fake hypervisor.

```
qemu: // / session
```

This connects you to the local URI QEMU-/KVM-Hypervisor (connection type *session*) is normal as a user.

```
qemu + unix: // / session
```

This connects you to the local URI QEMU-/KVM-Hypervisor (connection type *session*) is normal as a user.

```
qemu: // / system
```

This URI connects you as user *root* with the local QEMU-/KVM-Hypervisor (connection type *system*).

```
qemu + unix: // / system
```

This URI connects you as user *root* with the local QEMU-/KVM-Hypervisor (connection type *system*).

```
qemu + ssh: // / root@example.com / system
```

This connects you with a URI QEMU-/KVM-Hypervisor on the *example.com* node. The connection is made through an SSH tunnel.

```
xen: // /
```

This URI connects you with the local Xen hypervisor. This is the default setting.

```
vbox: // / session
```

This URI connects one as a normal user with the local VirtualBox hypervisor.

```
esx: // / example.com / no_verify = 1?
```

This URI connects you as user *root* to an ESX hypervisor on the *example.com* node. It is used for https protocol but not the certificate is verified.

```
gsx: // / example.com
```

This URI connects you as user *root* with a VMware server.

```
virsh # uri
```

The command returns the canonical *uri* URI of the hypervisor.

```
virsh # hostname
```

The *hostname* command displays the host name-Hypervisor.

```
virsh # NodeInfo
```

Displays information about the node. These include the number and type of CPUs and the amount of RAM.

```
virsh # capabilities
```

Returns an XML document that describes the properties of the currently connected hypervisors. This XML document also contains sections on skills of the host system and a set of descriptions for each supported host system. A detailed description can be found at the URL <http://libvirt.org/formatcaps.html>. To determine the KVM hypervisor checked, the driver of *libvirt* the directory */usr/bin* under the QEMU binaries and the device */dev/kvm*.

```
virsh # list [- inactive | - all]
```

Lists information about the domains. If no domain name is specified, information about all active domains output. With *-inactive*, all domains listed inactive. With *-all* are shown information for all domains. Here's an example:

```
virsh # virsh list
Id Name State
-----
0 Domain-0 running
2 fedora paused
```

In the column *ID* domains, the numeric IDs of the output. In *name*, the names of the domains. Under the conditions of the *State* domains are listed. There are six states:

r - running

The domain is ongoing.

b - blocked

The domain is blocked. That is, it does not work and can not be started. Perhaps the domain is waiting on input or expenditure (a typical case) or the domain is switched to the sleep state, as it has nothing else to do.

p - paused

The domain is paused by the *pause* command *xm*. In this state, it still consumes resources such as memory. The hypervisor can not take them into account when scheduling domain.

s - shutdown

The domain goes straight down. That is, the guest system has been the signal for the controlled shutdown received, but is still in the process of shutting down.

c - crashed

The domain has crashed. Normally occurs in this state if the domain is configured to not restart after a crash (see also *xmdomain.cfg*).

d - dying

The domain is in the process of dying. But it is not completely shut down or crashed.

```
virsh # FreeCell {cellno}
```

The *free* command displays the available free *cell* space for NUMA cell (non-uniform memory architecture). With *cellno*, the number of NUMA cell can be specified.

```
virsh # cpu-compare file
```

Compares a host CPU with a *file* in the XML file described CPU.

Commands to Manage Domains

The following commands manipulate domains directly. Most commands require as a first parameter is the *domain-id*. The *domain-id* UUID as a number (short integer), name or complete specified.

```
virsh # auto start [- disable domain-id]
```

Configures the domain that it is started automatically when the host system. The option *- disable* disables the automatic startup.

```
virsh # console domain-id
```

Connects the console to the virtual guest.

```
virsh # CreateFile
```

Starts a domain using the XML configuration file to *file*. This command starts the configured in the file domain, but the configuration is not stored in this file. The configuration of the domain is lost if the domain is shut down. The XML file for *virsh create* XML file must match the one that was generated *dumpxml* with the command.

```
virsh # define file
```

Defines a domain using the XML configuration *file* file. The domain will be created but not started. In contrast to the command *virsh create* the configuration persisted until the command is applied *undefine*.

```
virsh # destroy domain-id
```

Immediately terminates the domain *domain-id*. Here, the domain has no chance to respond and the host system is not finished properly. This is equivalent to unplugging the power cable at a real machine. In most cases it is better to use *shutdown* command.

```
virsh domblkstat # domain-id device
```

The command retrieves statistics *domblkstat* a block device from a running domain on. With *domain-id* UUID is the domain name, ID or the indicated. With the *block-device* Device is marked.

```
virsh domifstat # domain-id interface
```

The command retrieves statistics *domifstat* a network interface of a running domain from about. With *domain-id* UUID is the domain name, ID or the indicated. *Interface* with the network interface is marked.

```
virsh dominfo # domain-id
```

Prints a short information about the domain.

```
virsh domuuid # domain-id
```

Converts a domain name or domain ID to a domain UUID.

```
virsh DOMID # domain-name
```

Converts a domain name to a domain id using the internal mapping of the *libvirt* library.

```
virsh domname # domain-id
```

Converts a domain id to a domain name.

```
virsh domstate # domain-id
```

Returns the state of a running domain.

```
virsh # domain-id dump file
```

The *dump* command writes a dump of the core domain for analysis in the *file* file. For *domain-id* UUID is the domain name, ID or to specify.

```
virsh dumpxml # domain-id interface
```

Outputs a dump of the domain configuration as an XML file to standard output. The XML file can be used to *create* and *define* commands are used.

```
virsh # edit domain-id
```

The command *edit* to edit the XML configuration file of a domain. With *domain-id* UUID is the domain name, ID or the indicated. The selected editor is the environment variable \$ EDITOR set to. The *edit* command corresponds to the following sequence of commands:

```
~ # Virsh dumpxml domain> domain.xml  
~ # Edit domain.xml  
~ # Define virsh domain.xml
```

```
virsh # migrate [- live] domain desturi [migrateuri] [dname]
```

The *migrate* command moves a domain to another host. With - *live* is live migration enforced. With *domain* UUID is the domain name, ID or the indicated. The URI of the destination host is with *desturi* addressed. Optionally, the migration URI be defined with *migrateuri*. Does the hypervisor to rename the domain during the migration, the new name will be given by *dname*.

```
virsh # reboot domain-id
```

Starts a new domain. This corresponds to the *reboot* command in the console of the host system.

```
virsh # restore state-file
```

Represents the state of a domain from the *virsh save* file created with *Restore* (see command *save*).

```
virsh # save domain-id state-file
```

Saves the state of a running domain to a file. This state can later *restore* the command *virsh* to be restored. After saving is no longer and been assigned to the memory domain can be used by other domains. This corresponds approximately to the displacement of a real machine into sleep state with similar restrictions, such as time-outs for open network connections.

```
virsh schedinfo # domain-id [- set parameter = value] \  
[- Weight number] [- cap number]
```

The command scheduler *schedinfo* parameters are displayed or set.

domain-id

Domain Name, ID, or UUID.

- Set *parameter* = *value*

The parameter is passed by value.

- Weight *number*

Weight for XEN_CREDIT.

- Cap *number*

Cap for XEN_CREDIT.

```
virsh SETMEM # domain-id KiloBytes
```

Immediately changes the current memory allocation for the guest. The memory allocation is specified in kilobytes.

```
virsh setmaxmem # domain-id KiloBytes
```

Changes the maximum memory allocation for the guest. This value has no influence on the current memory usage. The maximum memory allocation is specified in kilobytes.

```
virsh setvcpus # domain-id count
```

Changes the number of virtual CPUs active (*count*) for the guest. The value of *count* can be limited by the host, the hypervisor or the original description of the domain.

```
virsh # shutdown domain-id
```

Is the host system, the signal for the correct shutdown. The host system must evaluate this signal correctly, to initiate a proper shutdown. There is therefore no guarantee that the shutdown occurs properly. Continue to need different times for different guest systems to shut down the services. For a Xen guest system is the parameter *on_shutdown* in the XML file with the domain's behavior on restart fixed.

```
virsh # start domain-id
```

The *start* command starts the defined, inactive domain *domain-id*.

```
virsh suspend # domain-id
```

Freezes a running domain. The allocated memory is preserved but the hypervisor into account when scheduling this domain any more. The command *resume* domain, the thawed.

```
virsh # resume domain-id
```

Ends the frozen state of the domain. The hypervisor into account when scheduling this domain again.

```
virsh ttyconsole # domain-id
```

The command is *ttyconsole* TTY device for the console. For *domain-id* UUID is the domain name, ID or to specify.

```
virsh # undefine domain-id
```

Deletes the configuration of an inactive domain. Since the domain is not running, UUID is the domain name or *domain* to be used for *id*.

```
virsh vcpuinfo # domain-id
```

Give brief information about the virtual CPUs in the domain. This includes information about the number of CPUs, the term and the memberships of the physical CPUs.

```
virsh vcpupin # domain-id vcpu cpuplist
```

Maps the virtual CPUs (*vcpu*) the physical CPUs on the host system. The parameter must be specified *vcpu cpulist*. Is a list of physical CPU numbers separated by commas are a respectively.

```
virsh vncdisplay # domain-id
```

Displays the IP address and port number of the VNC display. Such information is not available, issues an exit code of 1.

```
virsh # domxml-from-native format config
```

Converts a native host configuration to an XML-domain configuration. The source format is specified with the *format*. Conversion of options is QEMU *qemu* for the *use-argv*. The configuration data is defined with *config*. This option can be QEMU in a corresponding XML configuration file for *libvirt* convert. Here is a simple example:

```
Host ~ $ echo "/usr/bin/qemu-hda Platte.img>q.txt
Host ~ $ virsh domxml-from-native-argv q.txt qemu
```

```
virsh # domxml-to-native format xml
```

Converts an XML-domain configuration in a native host configuration. The target format is to *format* and XML file defines the *xml*. This can convert the settings from an XML configuration file into the appropriate QEMU options.

Commands to Manage Devices

The following commands associated to the domains devices are manipulated. The *domain-id* UUID as a number (short integer), name or complete specified. A documentation of the possible values and formats can be found at the URL <http://libvirt.org/format.html> (Device section).

```
virsh # attach-device domain-id file
```

Adds a device added to the domain. Here, the device definition is taken from an XML file (see documentation about the libvirt XML format).

```
virsh # attach-disk domain-id source target [- driver driver] \
[- Subdriver Subdriver] [- type type] [- mode mode]
```

Add a new disk device the domain added. The options are *source* and *target* paths to files and devices. For *drivers*, you can *file*, or inserting *physical tap*. This depends on the type of access. For *type* one uses an alternative to the default value or *floppy disk*, the values *cdrom* *mode* is. *Shareable* set or *read only*.

```
virsh # attach-interface domain-id type source [- target target] \
[- Mac] [- script script]
```

Adds a new network interface of the domain added. For *type* is either *network* (physical network device) or *bridge* (Bridge device used). The *source* option defines the source device and is *targeted* to the destination device in the host system specified. With *Mac* interface is a MAC address for the network specified. *script* with the option it is possible path to a script file specified. This can, in contrast to the default bridge, to build their own bridge.

```
virsh # detach-device domain-id file
```

Removes a device from the domain. The XML definitions correspond to those for the *attach-device*.

```
virsh # detach-disk target domain-id
```

Removes a disk device of the domain. For the *target* device is set up, which is the domain of looks.

```
virsh # detach-interface domain-id type [- mac mac]
```

Removes a network interface of the domain. For *type* is either *network* (physical network device) or *bridge* (Bridge device used). The *source* option defines the source device and is *targeted* to the destination device in the host system specified. When multiple network interfaces, it is recommended that the option *mac*'s MAC address that uniquely identifies the interface to pretend with.

```
virsh # nodedev-CreateFile
```

Generates a device on the physical host (node) with the data from the XML configuration file. This device can be assigned to a virtual machine. The XML configuration file *file* specified.

```
virsh # nodedev-destroy name
```

Clears the Device *name* on the physical host (node).

Commands for managing host Devices

```
virsh # nodedev-list [- cap string]
```

The command *nodedev-list* includes the devices that host on. With - *cap* the function name is specified.

```
virsh # nodedev-dumpxml device
```

The command is *dumpxml nodedev* of the node device in XML form to the standard output from the information. With the *device* key is the device specified.

```
virsh # nodedev-detach device
```

The *command-nodedev detach* removes a device driver from a node device before it is assigned a domain. With the *device* key is the device specified.

```
virsh # nodedev-reattach device
```

The *command-nodedev reattach* adds a device-driver device node added one before this one domain is assigned. With the *device* key is the device specified.

```
virsh # nodedev-reset device
```

nodedev-reset is the node device back before or after this one domain is assigned to the command. With the *device* key is the device specified.

Commands for managing storage pools

```
virsh # find-storage-pool-type sources [srcSpec]
```

The command *find-storage-pool-sources* identified potential sources for the storage pools. The output is a XML file. With *type sources*, the type of the search set. Optional file may be specified to search an XML *srcSpec* with.

```
virsh # find-storage-pool-sources-as type [host] [port]
```

The command *find-storage-pool-sources-as* will potential sources of storage pools. The output is a XML file. With *type sources*, the type of the search set. As an option to *host* a particular host and the port are examined *port*.

```
# virsh pool pool-autostart [- disable]
```

Configures the pool that it is started automatically when the host system. The option - *disable* disables the automatic startup. For UUID *pool* is the pool name or state.

```
virsh # pool-build pool
```

The *command-build* sets *pool* to a new pool. For UUID *pool* is the pool name or state.

```
# virsh pool CreateFile
```

The command *pool-create* generates a pool with the settings from the XML file, the *file*.

```
virsh # create-pool-as-name [- print-xml] type \
    [Source-host] [source-path] [source-dev] [source-name] \
    [Target]
```

The *command-pool create-generated* as a pool with the provisions of the options.

name

Name of the new pool.

- Print-xml

Prints the XML configuration. The pool will be defined or generated.

type

Type the new pool.

source-host

Source host for the underlying storage.

source-path

Source Path is the underlying storage.

source-dev

Source device for the underlying storage.

source-name

Source name for the underlying storage.

target

Target for the underlying storage.

```
virsh # define pool-file
```

The command *pool-define* defines a pool of the XML *file*. The pool is not started.

```
virsh # define-pool-as-name [- print-xml] type \
    [Source-host] [source-path] [source-dev] [source-name] \
    [Target]
```

The command *define-pool-defined* as a pool with the provisions of the options. The pool is not started.

name

Name of the new pool.

- Print-xml

Prints the XML configuration. The pool will be defined or generated.

type

Type the new pool.

source-host

Source host for the underlying storage.

source-path

Source Path is the underlying storage.

source-dev

Source device for the underlying storage.

source-name

Source name for the underlying storage.

target

Target for the underlying storage.

```
virsh # pool-destroy pool
```

The command *pool-destroy* destroy a pool. For UUID *pool* is the pool name or state.

```
virsh # pool-delete pool
```

The command *pool-delete* deletes a pool. For UUID *pool* is the pool name or state.

```
virsh # dumpxml pool pool
```

The *command-dumpxml pool* is a pool as an XML file to standard output from the information. For UUID *pool* is the pool name or state.

```
virsh # pool-edit pool
```

The command *pool-edit* to edit the XML file of a storage pool. For UUID *pool* is the pool name or state.

```
virsh # pool-info pool
```

The command *pool-info* provides information on a storage pool. For UUID *pool* is the pool name or state.

```
virsh # pool-list [- inactive] [- all]
```

The *pool command-list* is the list of active pools one of them. With *- inactive* pools are all the inactive list. With *- all* information will be displayed on all pools.

```
# virsh pool pool-name
```

The command converts a pool *pool-name-UUID* in a pool name.

```
virsh # pool-refresh pool
```

The *command-refresh* refreshed *pool* a pool. For UUID *pool* is the pool name or state.

```
virsh # pool-start pool
```

The *command-start pool* enabled the previously defined but inactive pool named *pool*.

```
# undefine virsh pool pool
```

The command deletes the configuration *pool* *undefine* an inactive pool. For UUID *pool* is the pool name or state.

```
virsh # pool-uuid pool
```

The *pool-uuid* command converts a pool name in a network UUID.

Commands for managing virtual disks

```
virsh # vol-createpool file
```

The command generated *vol-create* a volume *pool* called the settings *file* from the XML file.

```
virsh # vol-create-as poolname capacity [- allocation string] \
[- Format string]
```

The command *vol-create-generated* as a disk with the provisions of the options.

```
pool
```

Name of the pool.

```
name
```

Name of the disk.

```
capacity
```

Capacity of the disk. Optionally, the suffix k (kB), M (MB), G (GB) and T (TB) are given.

```
- Allocation string
```

Initially allocated size of the disk. Optionally, the suffix k (kB), M (MB), G (GB) and T (TB) are given.

```
- Format raw | bochs | qcows | qcows2 | vmdk
```

Sets the image format.

```
virsh # vol-delete [- pool string] vol
```

The *command-delete* deletes a disk *volume*. For UUID *pool* is the pool name or state. With *volume* path is the volume name, key, or defined.

```
virsh # vol-dumpxml [- pool string] vol
```

The *command-dumpxml* is about the volume *vol volume* as an XML dump to the standard output from the information. Desktop - UUID *pool* is the pool name or specify.

```
virsh # vol-info [- pool string] vol
```

The command *vol-info* is information on the volume of *vol*. Desktop - UUID *pool* is the pool name or specify.

```
virsh # vol-list pool
```

The command *vol-list* lists the volumes of the pool to *pool*.

```
virsh # vol-path [- pool string] vol
```

The command converts a *vol-path* volume UUID, a volume name or volume in a volume key path.

```
# virsh vol vol-name
```

The command converts a volume *vol-name-UUID*, Volume key, or volume path in a volume name.

```
virsh vol # vol-key
```

The command converts a *vol-key* volume UUID in a volume key.

```
virsh # vol-create-from pool file [- input string pool] vol
```

Generates a volume. In this case an existing volume is used as a template. *Pool* with the pool name is specified. *File* defines the XML configuration file. Optionally, with - *input* a name or a *pool* UUDI an input pools are defined. With *volume* key, the input volume or a given.

```
virsh # vol-clone [- pool <string>] vol newname
```

Clone a volume. With *volume* key, the source volume, or specified. The goal will be surrounded by *newname*. Optionally, with - a name or a *pool* UUDI a pool can be specified.

Commands for managing virtual networks

The following commands virtual networks can be manipulated. The library *libvirt* supports defining virtual networks, and the domains used by network devices can be connected with the real. Detailed information can be obtained at the URL <http://libvirt.org/formatnetwork.html>. Many of the commands for the manipulation of virtual networks are similar to the commands for manipulating domains. When naming virtual networks either the name or UUID is used.

```
virsh # net network-autostart [- disable]
```

Enables the automatic startup of the virtual network on boot. With the option - *disable* Autostart of the disabled.

```
virsh # net>CreateFile
```

Generates virtual network according to the XML *file* is a configuration file.

```
virsh # net-define file
```

Defines a virtual network according to the XML configuration *file* file. The network is defined but not enabled.

```
virsh # net network-destroy
```

Immediately deletes the specified virtual network. For *network* UUID is the name or the state.

```
virsh # net network-dumpxml
```

Provides information about the virtual network as an XML dump to the standard output.

```
virsh # net network-edit
```

Use the command *net-edit* you have to edit the XML configuration file for the virtual network *network*. The selected editor is the environment variable *\$ EDITOR* set to. The command *net-edit* matches the following command sequence:

```
~ # Virsh net dumpxml network> network.xml  
edit network.xml  
virsh define network.xml  
virsh # net-list [- inactive | - all]
```

Print a list of active networks. With *- inactive* networks are all inactive lists. With *- all* information will be displayed on all networks.

```
virsh # net network-name-UUID
```

Converts a network UUID to a network name.

```
virsh # net start-network
```

Start a defined but inactive network.

```
virsh # net-undefine network
```

Deletes the configuration of an inactive network.

```
virsh # network net-uuid-name
```

Convert a network name to a network UUID.

```
virsh # iface-list [- inactive] [- all]
```

Lists the physical interfaces on the host system. - With *- inactive*, the inactive and *all* interfaces are all listed.

```
virsh # iface interface-name
```

Converts a MAC address to an interface name. *Interface* with the MAC address is specified.

```
virsh # iface-mac interface
```

Converts an interface name to a MAC address. *Interface* with the interface name is specified.

```
virsh # iface dumpxml interface
```

Is interface information in XML format on standard output. *Interface* name with the MAC address or the specified interface.

```
virsh iface # define file
```

Defines a physical host interface from the XML *file* file. The interface is not started it.

```
virsh # iface undefine interface
```

Deletes the configuration of a physical host interfaces. *Interface* name with the MAC address or the specified interface.

```
virsh # iface-edit interface
```

The command is processed *iface-edit* the XML configuration file of a physical host interfaces. *Interface* name with the MAC address or the specified interface.

```
virsh # iface-start interface
```

Enables a physical host interface (*if-up*). *Interface* name with the MAC address or the specified interface.

```
virsh # iface interface-destroy
```

Disables a physical host interface (*if-down*). *Interface* name with the MAC address or the specified interface.

Commands for security

The following commands manipulate *secrets*. They are as passwords, digital keys and pass phrases. The library *libvirt* can save on their use *secrets* independently. Other objects, such as volumes, or domains, refer to the encryption on these *secrets*. *Secrets* are identified by UUID. More information can be found at the URL <http://libvirt.org/formatsecret.html>.

```
virsh # define secret-file
```

Generates a new *secret* based the specifications in the XML file to *file* without the associated value for the *secret*. If no UUID specified, a UUID is generated. If the XML file is a UUID of an existing use in *secret*, the existing properties are the *secrets* overwritten. The *secret* value is retained.

```
virsh # secret-secret dumpxml
```

Returns the properties of the specified *secrets* (UUID) XML dump to standard output as a.

```
virsh # secret-set-value secret base64
```

Sets the *secret* to (UUID) associated value. With the given value is *base64* (Base64 encoding).

```
# Secret-get-value secret virsh
```

To display the *secret* (UUID) associated value (encoded with Base64) on the standard output.

```
# undefine virsh secret-secret
```

Deletes a *secret* (UUID), including its value.

```
virsh # secret-list
```

View a list of UUIDs of the known *secrets* on the standard output.

Environment Variables

```
VIRSH_DEFAULT_CONNECT_URI
```

Defines the default URI to connect to the hypervisor. This option is the same format as used in the *connect*.

```
LIBVIRT_DEBUG = LEVEL
```

Sets the debugging level libvirt for all API calls. For more information on debugging options can be found at the URL <http://libvirt.org/logging.html>.

LIBVIRT_DEBUG = 1
 DEBUG level or higher.
LIBVIRT_DEBUG = 2
 Level **INFO** or higher.
LIBVIRT_DEBUG = 3
 WARNING level or higher.
LIBVIRT_DEBUG = 4
 ERROR level or higher.

See also

xm (1), xmdomain.cfg (5), xentop (1), <http://www.libvirt.org>

Authors

Andrew Puch <apuch@redhat.com>
Daniel Veillard <veillard@redhat.com>
Based on the XM-manpage
Sean Dague Dague at <sean@net>
Daniel at us dot ibm Stekloff <dstecklof@com>
Translation and addition: Robert Warnke, <http://qemu-buch.de>

COPYRIGHT

Copyright (C) 2005, 2007-2009 Red Hat, Inc.

LICENSE

virsh under the GNU LGPL v2 +.

Bugs

Bug reports can be used in the Red Hat Bugzilla site <https://bugzilla.redhat.com> be viewed here: https://bugzilla.redhat.com/buglist.cgi?/product=Fedora+Core+&component=&libvirt+bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&bug_status=MODIFIED&short_desc_type=allwordssubstr+short_desc=&long_desc_type=allwordssubstr

Referenced by

[virt-clone\(1\)](#), [virt-df\(1\)](#), [virt-install\(1\)](#), [virt-manager\(1\)](#), [virt-top\(1\)](#), [virt-viewer\(1\)](#)

virt-install**Name**

virt-install - Create new virtual machines (domains).

Synopsis

~ # **Virt-install [OPTION] ...**

Description

The command line tool **virt-install** is creating new virtual machines using libvirt hypervisor management library for. The tool supports both text based and the graphical installation via serial console, SDL or VNC. The host system, one or more virtual hard disks are made available. Virtual network interface of the virtual machines can be connected to the host system. The installation media can be integrated locally or remotely via NFS, HTTP and FTP. The tool **virt-install** files does the minimum needed to start the installation and allows the host system to access the rest of the distribution. With appropriate options allows **virt-install** the automatic installation of guest systems with a kick-start. The tool **virt-clone** systems can be cloned Guest existing, if automatic installation is not possible.

Options

If no options are given, ask **virt-install** from the corresponding values.

-H, --help

Prints a help and exit.

--Connect=CONNECT

Connects to a hypervisor by URI ago. Default connections based on the following rules:

xen

Applied to a host with a Xen kernel. It is to /proc/xen tested.

qemu: / / / system

If in a bare-metal kernel used as root.

qemu: / / / session

Applied to a bare metal kernel as unprivileged users. This URI is only required if it deviates from the default priority.

General

-N NAME, --name=NAME

If the new virtual machine a name. This must be unique. That is, a name can not be given more time. This also applies for inactive machines. If this option is not specified, ask **virt-install** it. For the new definition is the tool to drive down **virsh** system and the host to delete. Then, with the guest system is newly installed **virt-install**.

--RAM=MEMORY, --ram=MEMORY

With this option, the amount of RAM in MB for the guest system is defined. If the hypervisor does not have enough free memory, it is usually removed from the host system memory. If this option is not specified, ask **virt-install** it.

--Arch=ARCH

Is one different from the host system, processor architecture before. This option is only available for guest systems under QEMU available. The use of accelerators (KVM, KQEMU) is not possible. If this option is not used, the processor architecture of the host system is used.

-U UUID, - uuid = UUID

If the new virtual machine is a UUID. This is a hexadecimal number to use with 32 points. This UUID must be unique. That is, they can be awarded only once in the data center or in the world. This is important when manually setting the UUID. If no UUID specified, a random UUID will be generated.

- Vcpus = VCPUs

Defines the number of virtual CPUs for the new virtual machine. Not all support SMP hypervisor. In these cases, this value is ignored without notice.

- Check-cpu

Checks the number of virtual CPUs and the number of physical CPUs. If the number of virtual CPUs, the number of real CPUs, a warning is issued.

- Cpuset = cpuset

physical CPUs the guest system which may use Custom. *cpuset* is a list of numbers separated by commas are a respectively. It can also be specified from numbers-with a dash. Examples:

0,2,3,5

Use the processors 0,2,3 and 5

1-3,5,6-8

Use the processors 1,2,3,5,6,7 and 8

- Os-type = OS_TYPE

Optimizes the configuration of the virtual machine for the type of the host system. These include, inter alia, the optimal settings for ACPI and APIC and mouse drivers. Valid values for *OS_TYPE* are:

Linux (Linux 2.x)
Windows (Microsoft Windows 9x or higher)
unix (Traditional UNIX BSD or SysV derivatives)
other (operating systems that can not fit in any of the above groups.)

- Os-variant = OS_VARIANT

Allows a further optimization of the configuration of the virtual machine in addition to the - os-type. This option is not mandatory. Possible values are:

- Os-type = linux

rhel2.1 (Red Hat Enterprise Linux 2.1)
RHEL3 (Red Hat Enterprise Linux 3)
RHEL4 (Red Hat Enterprise Linux 4)
RHEL5 (Red Hat Enterprise Linux 5)
centos5 (Cent OS 5)
fedora5 (Fedora Core 5)
fedora6 (Fedora Core 6)
fedora7 (Fedora 7)
SLES10 (Suse Linux Enterprise Server 10.x)
debianEtch (Debian 4.0 Etch)
debian etch (Debian Lenny)
generic26 (Generic Linux 2.6.x kernel)
generic24 (Generic Linux 2.4.x kernel)

- Os-type = windows

winxp (Microsoft Windows XP)
win2k (Windows 2000)
win2k3 (Microsoft Windows 2003)
vista (Windows Vista)

- Os-type = unix

solaris9 (Sun Solaris 9)
solaris10 (Solaris 10)
freebsd6 (FreeBSD 6.x)
openbsd4 (Open BSD 4.x)

- Os-type = other

msdos (Microsoft DOS)
netware4 (Novell Netware 4)
NetWare5 (Novell Netware 5)
netware6 (Novell Netware 6)

- Host-device = HOSTDEVS

Bind physical devices to the host domain.

Full virtualization

- Sound

Enables emulation of sound devices.

- Noapic

Disabled APIC in host systems with full virtualization. Here, the setting - os-variant and - os-type overwritten.

- Noacpi

Disable ACPI in host systems with full virtualization. Here, the setting - os-variant and - os-type overwritten.

Virtualization species

-V, - hvm

Identifies the host system as a fully virtualized guest. If both para-and full-virtualization is available with this option, the full virtualization is enforced. This parameter is not in the host systems without hardware support used. This option is available when connecting to one based on QEMU hypervisor.

-P, - paravirt

Identifies the host system as a paravirtualized guest. If both para-and full-virtualization is not available and the option - paravirt nor - given hvm is a Virtualisierungsart asked

for.

- Accelerate

Enables a kernel accelerator (KQEMU or KVM) for the QEMU guest. This option should always be used in QEMU / KVM, unless the host system is not compatible with kernel accelerators. If both the KVM and KQEMU are available, use KVM. QEMU version 0.12.0 does not support more KQEMU.

Installation methods

-C cdrom, - cdrom = CDROM

Refers to a file to be used as a virtual CD / DVD-ROM drive. This can be a path to an ISO image or a real CD / DVD-ROM device. The ISO image can be addressed via a URL. The URL has the same format as the - location. If - cdrom not specified, the - location or - pxe necessary.

LOCATION-1, - location = LOCATION

Defines the installation source. For paravirtualized guest systems have this source the kernel and initial RAM disk (initrd) included. For fully-virtualized guest systems can also - location source kernel and initial RAM disk to be a reference to. Alternatively, with - cdrom ISO image to install specified. LOCATION has the following formats:

DIRECTORY

Path to a local directory with the installation media.

nfs: host: / path, nfs: / / host / path

URL of an NFS server with the installation media.

http://host/path

URL of an HTTP server using the installation media.

ftp://host/path

URL of an FTP server with the installation media.

- Pxe

It is used for the PXE boot protocol to load the initial RAM disk and the kernel to start the installation of the host system. If the option - pxe is not specified, the - location or - cdrom necessary.

- Import

Generates the domain using an existing virtual disk.

- Livecd

Identifies the installation media as a live CD. In order to boot from CD / DVD drive is set permanently. Usually this option is the option - used nodisks.

-X EXTRA, - extra-args = EXTRA

Specifying additional kernel options for the installation kernel and initial RAM disk (optional - location). These are passed to the installer of the host system.

Storage configurations

- Disk = DISKOPTS

Defined virtual storage for the guest. This option replaces the - file, - file-size and - nonsparse. The general format - disk opt1 = val1, val2 = opt2 ... To define a virtual storage medium, the following options are required:

path

Defines the path to an image file or block device for use as a virtual storage medium for the guest. This path does not exist or the file, a virtual disk is created. This is also the option to specify the size of size. If this memory device to another computer, it must be released as libvirt storage volume. Does the home directory path to a libvirt storage pool on the host, the new storage is as libvirt storage volume created. remote computer must be specified for the base directory for access to the storage pool.

pool

Specifies the name of an existing libvirt storage pool for the generation of new storage. Required option is the size.

vol

Specifies to use the libvirt storage pool. This is poolname / volname be expressed as.

Other options:

device

Defines the type of the device. Possible values are cdrom, disk (default) or floppy. If no option is specified and installation method uses the cdrom is used as an installation media cdrom.

bus

Defines the type of the bus. Possible values are ide, scsi, usb, or xen virtio. The default setting depends on the hypervisor.

perms

Sets the access rights. Possible values are RW (Read / Write), RO (Read Only), or sh (Shared Read / Write). The default value is rw.

size

Defines the size of the new virtual disk. The size of the new virtual disk is given in GB.

sparse

Possible values are true (default) or false. Usually the size of the virtual hard disk in the file system of the host only about their level. That is, the size of the image file grows with the data that is stored in this virtual hard disk. So when creating the image file, whose full size is allocated option is to apply false. Creating the image file takes a little more time here. Therefore, a speed gain results in the access by the host system. Furthermore, one avoids I / O error in the host system if the virtual hard disk does not fit into the file system of the host.

DISK-f FILE, - file = FILE DISK

Defines the path to an image file, a disk partition or a logical volume for use as a virtual storage medium for the guest. This path does not exist or the file, a virtual disk is created. This is in addition to the - file-size the size of pretending. The-f or - file can be specified multiple times to add multiple virtual disks. If this option is not specified, ask virt-install it. This option is the - = DISKOPTS replaced disk.

DISK SIZE-s, - file-size = DISK SIZE

Defines the size of the new virtual disk. For a virtual disk is created, is also the *option-file* needed, using the specified file does not exist. The size of the new virtual disk is given in GB. This fractional numbers are possible. The memory requirements for the new file will only be reserved if the - specified *nonsparse*. If the option - *file-size* is not specified, then asks *virt-install* and puts new file to a. This option is the - = *DISKOPTS* replaced *disk*.

- Nonparse

Usually the size of the virtual hard disk in the file system of the host only about their level. That is, the size of the image file grows with the data that is stored in this virtual hard disk. So when creating the image file, whose full size is allocated, the option - to apply *nonsparse*. Creating the image file takes a little more time here. Therefore, a speed gain results in the access by the host system. Furthermore, one avoids I / O error in the host system if the virtual hard disk does not fit into the file system of the host. This option is the - = *DISKOPTS* replaced *disk*.

- Nodisks

Does the new virtual machine no disk, then the option - specify *nodisks*. This is (iSCSI or NFS root), for example in live CDs or install on network storage need. This option disables all the demands for managing virtual disks.

Network Configuration

- B BRIDGE, - bridge = BRIDGE

Defines a bridge device to connect to the network card of the host system. This option is no longer use (*deprecated*). Instead, the option - to apply *network*.

- W NETWORK, - network = NETWORK

Connects the host system to the network of the host system. This option can be specified multiple times to emulate multiple virtual NICs. If this option is not specified, created for the host system on a network card. Does the system host a bridge device with a physical network card as a slave, it is used for the connection. Failing this, the virtual network used the *default* name with. *NETWORK*, the following formats accept a.

- bridge: BRIDGE

Connects to the host system in the existing bridge called *the Bridge*. This option is used if the network configuration is static in the host and the guest system requires both an unrestricted and LAN connections. Furthermore, this option is applied when a live migration of virtual machine is provided.

- network: NAME

Connects to a virtual network in the host named her *name*. Virtual networks can command line tool *virsh*, available to be created and deleted. In a no-install *libvirt* there is usually a virtual network with the *default* name. Does the system host a dynamic network configuration, for example by the *network manager*, or it uses a wireless LAN is a virtual network --*network* = *network*: use *NAME*. The host system receives a NAT address outside of the host system when the connection is.

- user

Connect to the LAN with SLIRP ago. This is a restricted version of NAT. This option applies only for QEMU instances that are operated under a unprivileged users.

- M MAC, - mac = MAC

Sets a fixed MAC address for the network interface of the host system. If this option is not specified or the value - = *RANDOM* used *mac*, *MAC* is a random value is calculated. For Xen-guest systems e are the first three pairs of values equal to *00:16:3* set. With virtual machines under QEMU or KVM, however, is to specify *54:52:00*.

- Nonetwork

It is provided to the host system is not a network.

Configurations for graphics output

- Vnc

Generates a virtual console in the guest system and export them as a VNC server on the host. If no port with the - set *vncport*, the VNC server is the first free port from 5900 to connect. The current classification of the VNC display is determined by the command *virsh vncdisplay*. If neither this option nor the - *sdl* or - given *nographics* asks *virt-install* after VNC configuration.

- Vncport = VNCPORT

Sets the permanent, static port number for the VNC console of the host system. When assigning the port number is important to note that there is no conflict with VNC consoles are other guest systems.

- Sdl

Generates a virtual console in the guest system and renders the output to an SDL window in the host system. If the SDL window is closed, the virtual machine is terminated uncontrollably.

- Nographics

Disables the graphic output of the virtual console of the host system. A text-based console is still on the first serial port or a paravirtualized device consoles available.

- Noautoconsole

Disable automatic connection to the console of the host system. The normal behavior is about connecting to the graphical console with a VNC client or a text-based console with the command *console virsh -*. With the option *noautoconsole* this behavior is disabled.

- KEYMAP-k, - keymap = KEYMAP

Defines the keyboard layout. By default the English keyboard is set.

Other Options

- D, - debug

Output of debugging information to the terminal during the installation process. This debugging information is independent from this option in the file \$ *HOME* / .*virtinst* / *virt-install.log* written.

- Noreboot

Disable the automatic restart of the graphical installation.

- Wait = WAIT

Defines the waiting time in minutes for the complete installation of the host system. Without this option is waiting on the *virt-install* the panel. If - *noautoconsole* used *virt-install* the complete installation of control. A value of 0 has the same effect as - *noautoconsole*. If the waiting time is exceeded, *virt-install* completed.

- Force

Answered all appropriate questions from *virt-install* with *yes*. All other questions will be terminated.

- Prompt

Forces a user's input. The default setting is *false*.

Examples

In this example, a KVM guest system is installed. Here, a new image is created using a virtual network and the installation of the host system from the CD / DVD-ROM drive of the host system starts. Furthermore, the VNC server is activated.

```
~ # Virt-install \
- Connect qemu: / / / system \
- Name demo \
- Ram 500 \
- Disk path = / var / lib / libvirt / images / demo.img, size = 5 \
- Network network: default \
- Accelerate \
- Vnc \
- Cdrom / dev / cdrom
```

In this example, Fedora 9 installed as a KVM guest system. The storage medium is an LVM partition. It uses a virtual network and booted via PXE. Furthermore, the VNC server is activated.

```
~ # Virt-install \
- Connect qemu: / / / system \
- Name demo \
- Ram 500 \
- Disk path = / dev / HostVG / DemoVM \
- Network network: default \
- Accelerate \
- Vnc \
- Os-variant fedora9
```

In this example, a QEMU guest with a PPC architecture on a real partition of the host system is installed. This SDL graphics is used. The boot process is performed by an external kernel and an initial RAM disk.

```
~ # Virt-install \
- Connect qemu: / / / system \
- Name demo \
- Ram 500 \
- Disk path = / dev / hdc \
- Network bridge: eth1 \
- Arch ppc64 \
- Sdl \
- Location http://archives.fedoraproject.org/pub/archive/fedora/linux/core/6/ppc/os/
```

In this example, boot with the Xen fully virtualized guest system from an image of a live CD with no hard drives.

```
~ # Virt-install \
- Hvm \
- Name demo \
- Ram 500 \
- Nodisks \
- Livecd \
- Vnc \
- Cdrom / root/fedora7live.iso
```

Installing a paravirtualized Xen guest system with 500 MB of RAM and a 5 GB hard drive: The host system is Fedora Core 6 installed on a Web server. Installation is done in text mode. It is the legacy - file used.

```
~ # Virt-install \ - paravirt \ - name demo \ - ram 500 \ - file / var / lib / xen / images / demo.img \ - file-size 6 \ - nographics \ - location http://
```

Authors

Written by Daniel P. Berrange and Hugh Brock, Jeremy Katz, Cole Robinson and a team. See the AUTHORS file in the sources.

Translation and addition: Robert Warnke, <http://qemu-buch.de>

Bugs

Bugs to the mailing list <http://www.redhat.com/mailman/listinfo/et-mgmt-tools> or directly to BugZilla <http://bugzilla.redhat.com/bugzilla/> (Fedora, python virtinst) to . Report

Copyright

Copyright (C) 2006-2007 Red Hat, Inc. and other authors. This is free software (GNU General Public License, <http://www.gnu.org/licenses/gpl.html>).

See also

[virsh](#) (1), [virt-clone](#) (1), [virt-manager](#) (1), <http://virt-manager.org>

Referenced by

[virt-image](#) (1), [virt-image](#) (5)

virt-clone

Name

[virt-clone](#) - Clone an existing virtual machine.

Synopsis

```
~ # Virt-clone [OPTION] ...
```

Description

[virt-clone](#) command line tool is a clone virtual machines using *libvirt* to the library. Here, the disk image copy of an existing virtual machine and generates a new guest system with identical hardware configuration. Elements that must be unique and unambiguous, such as Mac addresses can be modified to avoid conflict. With these options works [virt-clone](#) without manual intervention. Kick-start procedures, the guest systems are supported.

Options

If neither of these options are given, the necessary values can be queried interactively.

```
-H, - help
```

Displays a help and exit.

```
- Connect = CONNECT
```

Connects to a hypervisor by URI ago. Default connections based on the following rules:

```
xen
```

Applied to a host with a Xen kernel. It is to `/proc/xen` tested.

`qemu: / / system`

If in a bare-metal kernel used as root.

`qemu: / / session`

Applied to a bare metal kernel as unprivileged users. This URI is only required if it deviates from the default priority.

General

`ORIGINAL_GUEST-o, - original = ORIGINAL_GUEST`

Name or UUID of the host system to dublizieren. The host system must be shut down for cloning and off.

`- Original-xml = ORIGINAL_XML`

The XML configuration file `ORIGINAL_XML` is used as the source.

`NEW_NAME-n, - name = NEW_NAME`

Name the new virtual machine. This must be unique. That is, a name can not be given more than once. This also applies to all inactive machines. If this option is not specified, then asks `virt-clone`. For the new definition is the tool to drive down `virsh` system and the host to delete. Then `clone` will be the guest system using `virt-duplicated`.

`NEW_UUID-u, - uuid = NEW_UUID`

If the new virtual machine is a UUID. This is a hexadecimal number to use with 32 points. This UUID must be unique. That is, they can be awarded only once in the data center or in the world. This is important when manually setting the UUID. If no UUID specified, a random UUID will be generated.

Storage configurations

`DISK-f FILE, - file = FILE DISK`

Defines the path to an image file, a disk partition or a logical volume for use as a virtual storage medium for the guest. If the source instance of multiple virtual disks, this option must be specified more than once. If this option is not specified, then asks `virt-clone`.

`- Force-copy = TARGET`

Forced copying devices. If, for example `hdc` read-only CD-ROM drive, so with `-copy = hdc` be forced to `copy-force`.

`- Nonsparse`

when creating the image file for the target instance is already reserved their full size. This is the option `-apply nonsparse`. Creating the image file takes a little more time here. Therefore, a speed gain results in the access by the host system. Furthermore, one avoids I/O error in the host system if the virtual hard disk does not fit into the file system of the host.

`- Preserve-data`

Keep a file for use as an image for the new instance.

Network Configuration

`-m NEW_MAC, --mac=NEW_MAC`

Sets a fixed MAC address for the network interface of the host system. If this option is not specified or the value `- = RANDOM` used `mac`, `MAC` is a random value is calculated. Für Xen-Gast-Systeme sind die ersten drei Wertepaare gleich `00:16:3e` zu setzen. Bei virtuellen Maschinen unter QEMU oder KVM ist dagegen `54:52:00` vorzugeben.

Other Options

`-D, - debug`

Output of debugging information to the terminal during the installation process. This debugging information is independent from this option in the file `$HOME/.virtinst/virt-install.log` written.

`- Prompt`

Forces a user's input. The default setting is `false`.

`- Force`

Answered all appropriate questions from `virt-install` with `yes`. All other questions will be terminated.

Examples

Klonnt eine virtuelle Maschine mit dem Namen `demo`, die eine einzige virtuelle Festplatte besitzt.

```
~# virt-clone \
--original demo \
--name newdemo \
--file /var/lib/xen/images/newdemo.img
```

Klonnt ein QEMU-Gast-System mit mehreren virtuellen Festplatten.

```
~# virt-clone \
- Connect qemu: / / system \
--original demo \
--name newdemo \
--file /var/lib/xen/images/newdemo.img \
--file /var/lib/xen/images/newdata.img
```

Klonnt eine virtuelle Maschine auf ein physikalisches Device, welches mindestens die gleiche Größe besitzt wie die originale virtuelle Festplatte des originalen Gast-Systems. Ist das Ziel-Device größer als das Quell-Device, kann das neue Gast-System sein Datei-System vergrößern.

```
~# virt-clone \
- Connect qemu: / / system \
- Name demo \
--file /dev/HostVG/DemoVM \
--mac 54:52:00:34:11:54
```

Authors

Geschrieben von Kazuki Mizushima und einem Team aus vielen anderen Autoren. Siehe Datei AUTHORS im Quell-Code für weitere Details.

Übersetzung: Robert Warnke, <http://qemu-buch.de>

Bugs

Bugs sind an die Mailing-List <http://www.redhat.com/mailman/listinfo/et-mgmt-tools> oder direkt an BugZilla <http://bugzilla.redhat.com/bugzilla/> (Fedora, python-virtinst) zu melden.

Copyright

Copyright (C) Fujitsu Limited 2007 und weiteren Autoren. This is free software (GNU General Public License, <http://www.gnu.org/licenses/gpl.html>).

See also

[virsh\(1\)](#), [virt-install\(1\)](#), [virt-manager\(1\)](#), <http://virt-manager.org>

virt-viewer**Name**

virt-viewer – Anzeige der grafischen Konsole einer virtuellen Maschine.

Synopsis

```
~# virt-viewer [OPTIONS] domain-id | id | uuid
```

Description

virt-viewer ist ein kleines Tool zum Anzeigen der grafischen Konsole einer virtuellen Maschine unter Verwendung des VNC-Protokolls. Die virtuelle Maschine wird über seinen Namen, seine ID oder seine UUID adressiert. Wenn die virtuelle Maschine nicht läuft, kann der **virt-viewer** so konfiguriert werden, dass er auf den Start der Instanz wartet. Das Tool **virt-viewer** kann sich zu einem entfernten Host verbinden.

Options

Die folgenden Optionen sind möglich:

-H, **--help**

Prints a help.

-V, **--version**

Zeigt die Version des Programms an.

-v, **--verbose**

Zeigt Informationen über die Verbindung an.

-c URI , **--connect=URI**

Definiert die URI des Hypervisors, zu dem die Verbindung aufgebaut werden soll.

-w, **--wait**

Wait boots with the connection to the console of an inactive virtual machine to the host system.

-D, **--direct**

Uses an SSH tunnel to connect to the console, even if the URI uses the main connection, the SSH protocol.

Examples

Connects to the Xen host system named her *demo*.

```
~ # Virt-viewer demo
```

Connects to a QEMU guest system with the ID 7 ago.

```
~ # Virt-viewer - connect qemu: / / / system 7
```

Connects to a host system with the specified UUID ago. It will wait for the connection, are the guest system is booted.

```
~ # Virt-viewer - wait 66ab33c0-6919-a3f7-e659-16c82d248521
```

Connects to a remote console using TLS (Transport Layer Security) ago.

```
~# virt-viewer --connect xen://example.org/ demo
```

Stellt eine Verbindung zu einer Remote-Konsole unter Verwendung von SSH her. Danach erfolgt der Verbindungsaufbau zu dem Gast-System über dem aufgebauten SSH-Tunnel.

```
~# virt-viewer --connect xen+ssh://root@example.org/ demo
```

Authors

Geschrieben von Daniel P. Berrange, basierend auf dem GTK-VNC-Beispiel-Programm *gvncviewer* .

Übersetzung: Robert Warnke, <http://qemu-buch.de>

Bugs

Bugs sind an die Mailing-List <http://www.redhat.com/mailman/listinfo/et-mgmt-tools> zu melden.

Copyright

Copyright (C) 2007 Red Hat, Inc. und weitere Autoren. This is free software (GNU General Public License, <http://www.gnu.org/licenses/gpl.html>).

See also

[virsh\(1\)](#), [virt-manager\(1\)](#), <http://virt-manager.org>

virt-manager

- <http://virt-manager.org>

Name

virt-manager – Desktop-Tool zum Managen von virtuellen Maschinen.

Synopsis

```
~ # Virt-manager [OPTIONS]
```

Description

virt-manager is a desktop tool for managing virtual machines. It enables the control of virtual machines, generating new virtual machines, manage virtual networks, access to the

graphical console of virtual machines and display of performance statistics.

Options

The following options are supported:

-H, - help

Displays a help.

- Profile = FILE

Generates runtime performance profiles and stores them in the file *FILE*.

URI-c, - connect = URI

Defines the URI to connect to the hypervisor.

- Debug

Returns debugging information on standard output. In addition, the option - no-fork indicated.

- No-dbus

Disables the DBus API for remote control of user interface from the *virt-manager*.

- No-fork

Prevent the diversion into the background.

- No-conn-autostart

Suppresses automatic connections.

- Show-domain-creator

Starts the wizard for creating new virtual machines.

- Show-domain-editor = UUID

Display the dialog box to edit the properties of the virtual machine with the specified UUID.

- Show-domain-performance = UUID

Display the dialog box to monitor the performance of the virtual machine with the specified UUID.

- Show-domain-console = UUID

View the virtual console of the virtual machine with the specified UUID.

- Show-host-summary

Display the dialog box to monitor the performance of all virtual machines.

Authors

Written by Daniel P. Berrange.

Translation and addition: Robert Warnke, <http://qemu-buch.de>

Bugs

Bugs must be reported to the mailing list <http://www.redhat.com/mailman/listinfo/et-mgmt-tools>

Copyright

Copyright (C) 2006-2007 Red Hat, Inc. and other authors. This is free software (GNU General Public License, <http://www.gnu.org/licenses/gpl.html>).

See also

[virsh \(1\)](#), [virt-viewer \(1\)](#), <http://virt-manager.org>

Referenced by

[virt-clone \(1\)](#), [virt-install \(1\)](#)

[<<< | # # | >>> http://qemu-buch.de](http://qemu-buch.de)

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_libvirt"

This page has been accessed 13,699 times. This page was last updated on 17 October 2010 at 17:12 clock changed. Content is available under GNU Free Documentation License 1.2 .

Simple Protocol for Independent Computing Environments, Spice Client

(Link to this page as [[QEMU-KVM-book / notes / Spice]])

<<< | # # # | >>> | English

Contents

- 1 Spice
 - 1.1 Client
 - 1.1.1 Syntax
 - 1.1.2 Command Options
 - 1.1.3 Keyboard shortcuts

Spice

Website: <http://www.spice-space.org>

The *Simple Protocol for Independent Computing Environments* (Spice) is a software and a protocol to support access to remote machines. It offers good multimedia support. Media Streaming and VoIP is possible. Spice is a multi-tier architecture, server, client and a driver. On the server side is emulated by a VDI device in the host system, a PCI device. On the client side can be either a browser or a dedicated client application is used. Is being developed by the company Red Hat Spice, Inc. as open source. There is an implementation for QEMU / KVM.

Client

Syntax

```
~ $ Spicec-h host [-p port] [-s secure-port] [options]
```

Command Options

-H, - host host

Defines the address of the Spice server.

-P, - port port

Defines the port of the Spice server.

-S, - secure-port port

Defines the port of the Spice Secure server.

- Secure-channels ch0, ch1 ...
- Unsecure channels ch0, ch1 ...

Used by Spice transmission channels (channels) can be secured or unsecured. With the *- secure-encrypted channels*, certain channels. With the *- unsecure channels*, certain channels unencrypted. There is the Spice channels *main, display, input, cursor, play back and record*. The default is supported for all channels of both the secured and the unsecured mode. The relevant use depends on the applied options *- secure-port*, and *- from port*.

-W, - password password

Sets the ticketing password. By default, no password is defined.

f, - full-screen [= auto-conf]

Spice opens the client in full screen mode. Optionally, the automatic configuration with *auto-conf* enabled. Here, the remote display is adapted to the display of the Spice clients. For this, a host agent installed and running is necessary.

- Canvas-type *type1*, *type2* ...

With this option selected, one or more renderers. If more than one renderer given, specifies the order priority. For Linux clients is only available to *cairo*. For Microsoft Windows clients and can be used *cairo gdi*. The default value is *gdi*.

- Enable-channels *ch0*, *ch1* ...

Activates the specified channel. With *all* (default), all available channels enabled. Possible channels are: *display*, *input*, *cursor*, *play back* and *record*.

- Disable-channels *ch0*, *ch1* ...

Disables the specified channel. With *all* channels are all available off. Possible channels are: *display*, *input*, *cursor*, *play back* and *record*. Per Default all channels are enabled.

- Help

Prints a help.

Keyboard Shortcuts

[Shift] + [F11] Toggle full screen mode on or off.

[Shift] + [F12] Frees the cursor when it is in the window.

In debug mode also apply these keys:

[Shift] + [F5] Connects to the server.

[Shift] + [F6] Disconnect from the server.

<<< | # # # | >>> http://qemu-buch.de

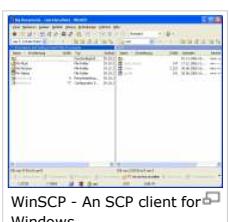
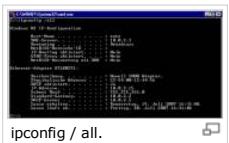
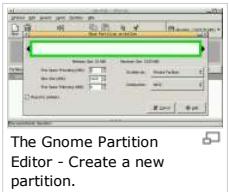
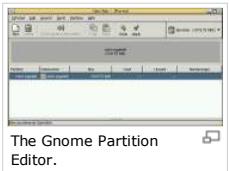
Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Spice "

This page has been accessed 2693 times. This page was last updated on 27 September 2010 at 06:34
clock changed. Content is available under GNU Free Documentation License 1.2 .

Toolbox PowerShell Shell Cygwin ssh netcat download bzip2 install 7-Zip RAR chmod chown sudo fdisk mount runas dd dcfldd SSHFS TFTP Synergy dnsmasq

(Link to this page as [[QEMU-KVM-Buch / Notes / Useful Tools]])

<<< | # # # | >>> | English



Contents	
1 Useful Tools	
1.1 Command Line Interpreter	
1.1.1 Unix shells	
1.1.2 DOS Command Prompt	
Cygwin 1.1.3 - Unix tools for Microsoft Windows	
1.1.4 Windows PowerShell	
1.2 Help positions	
1.3 File and Directory Commands	
4.1 Links	
1.5 determine file type	
1.6 Text Files	
1.7 Text files edit	
1.8 add files to each other	
1.9 Compression and Archiving	
ZIP 1.9.1	
1.9.2 GZIP	
Bzip2 1.9.3	
1.9.4 tar	
1.9.5 7-Zip	
RAR 1.9.6	
1:10 user and rights management	
1.10.1 Unix / Linux	
Microsoft Windows 1.10.2	
1:11 media	
1.11.1 dd dd_rescue and dcfldd	
1.11.2 Partitioning	
1.11.3 Mounting File Systems	
1.11.4 Level of partitions determine	
1:12 Network configuration and test	
1:13 network services and clients	
1.13.1 SSH	
1.13.2 putty	
1.13.3 SCP	
1.13.4 WinSCP	
1.13.5 SSHFS	
1.13.6 FTP	
1.13.7 TFTP	
Wget 1.13.8	
1.13.9 Telnet	
1:13:10 netcat	
1:13:11 TightVNC	
1:13:12 Remote Desktop Protocol	
1:13:13 Synergy	
1:13:14 dnsmasq	
1:14 Compile	



Useful Tools

In this section some useful tools are explained. This brief review serves to get along with less well-known guest systems. Most of the tools described here are to serve as a command-line commands, and come from the Unix environment. They are often also for other platforms such as Microsoft Windows versions available (see <https://ninite.com>).

Command Line Interpreter

Unix-Shells

Unix servers are usually operated via the command line. This is because the shell (command line interpreter) are very powerful on Unix. In addition, a shell is superior in some tasks of a graphical user interface, such as flow control and program responses may be included. In addition to the interactive use of the shells have to perform the task shell scripts. There are Unix / Linux different shells available, such as *sh*, *csh*, *ksh*, *bash*, *tcsh* and *zsh*. On Linux, *bash* is used frequently.

DOS Command Prompt

Under the old DOS-based versions of Microsoft Windows 95, 98 and ME *command.com* was used to provide a DOS prompt. Microsoft Windows NT and successors and share new, largely backward-compatible command-line shell called *cmd.exe* one. The start is the DOS command prompt in newer versions of Microsoft Windows on any item on the Start menu. In the appearing command-line specifies the command *cmd*. To finish the DOS prompt with *exit*.

```
C: \> exit
```

Cygwin - Unix tools for Microsoft Windows

Download: <http://www.cygwin.com/setup.exe>

Cygwin is an emulation of the Linux API on various versions of Microsoft Windows. This allows programs that usually run on POSIX systems such as GNU / Linux, BSD and Unix, ported to Microsoft Windows. POSIX (Portable Operating System Interface) is a Unix developed for standardized *application programming interface*. To install you download the setup program and start it. This small program, all necessary files obtained directly from different FTP servers. During installation, three options are offered.

Install From Internet The required files are downloaded from the Internet during installation.

Download without installing All necessary files are downloaded for later installation but not installed.

Install from Local Directory The files necessary for installation previously on the option *Download without installing* downloaded already.

For a one-time installation, select *Install From* is to use the *Internet*. In the next dialog, the list is selected to be installed in the Cygwin. The default for this is *C: \cygwin*. Furthermore, it is queried whether the logged or all users can start Cygwin, recommend, to all users (*All Users*) to allow the use of Cygwin. It will be asked to interpret the format in which text files to be. Here is the format recommended *Unix*. Then the directory for the files is downloaded to pretend. The next step will be prompted for a proxy server. If no proxy is available, the option *Direct Connection* is the right choice. When selecting a mirror server, it is advisable to select a server close by. The setup program will download a list of available packages and offers them for selection. In the default setting, the *bash* shell, and some additional tools are installed. At a minimum the package groups *Archives*, *Doc*, *net*, *system*, *text* and *web* are selected for installation. To do this click on the list next to the name of the package group to the *default* text. Instead *Default Install* must be in each case. Packages can subsequently be installed by calling *setup.exe* again, too. After clicking the *Next* button Cygwin start the download and installation of selected packages. At the end of the installation is the query whether entries in the start menu and an icon to be set up on the desktop. Cygwin is started by clicking the icon. Cygwin provides a Unix environment. On Unix the path separator is a slash (/) and not a backslash (\). The Windows Registry is Cygwin under */proc / registry* scans as. Drive letters (C:, D:, ...), as in Microsoft Windows are not available. For example, from the Windows directory *C: \ Temp test.txt* in the Cygwin path */cygdrive / C / temp / . Test.txt* The root directory (/) corresponds to the Cygwin Cygwin installation path. If Cygwin in *C: \ cygwin* is installed, the two paths */cygdrive / C / usr / bin / gzip.exe* and */usr / bin / gzip.exe* equivalent. A change of the drive is *cd* followed by the drive letter. For example, the following command changes to the C drive.

```
~ \$ cd c:
```

Windows PowerShell

Download: <http://www.microsoft.com/germany/technet/scriptcenter/hubs/msh.mspx>

A comfortable shell under Microsoft Windows, the Windows PowerShell. PowerShell is a modern alternative to the Windows command line utility for *cmd.exe* and Windows Script

Host.

help

On Unix / Linux or Cygwin you get help on the manual pages. It is called by the man command followed by the command that you want to have explained. Example:

```
~ $ Man qemu
```

If the command is unknown, apropos to list all the manual pages for a specific keyword with.

```
~ $ Apropos qemu
qemu (1) - QEMU Emulator User Documentation
qemu-doc (1) [qemu] - QEMU Emulator User Documentation
qemu-img (1) - QEMU disk image utility
qemu (1) - QEMU Emulator User Documentation
qemu-doc (1) [qemu] - QEMU Emulator User Documentation
qemu-img (1) - QEMU disk image utility
```

On Microsoft Windows in the DOS window with the help command followed by the command that you want to have explained, help show a. The following example shows a command to help you.

```
C: \> help you
```

On Microsoft Windows, there is still an aid in the Start menu to Help and Support.

File and Directory Commands

To change the directory used on Unix / Linux and Microsoft Windows command cd followed by the name of the destination directory.

```
~ $ Cd directory
```

A directory higher you get with cd followed by two points. On Unix / Linux must include a space after cd ever.

```
~ $ Cd ..
```

To determine the current directory path is Unix / Linux under the command pwd. On Microsoft Windows can be applied to the dir command. This lists the contents of the current directory.

```
C: \> dir
```

On Unix / Linux command is used to list the contents of ls. With the -l option, a detailed display. The following information is output to the file:

- File Rights
- Ownership
- Group membership
- Size
- Last modified
- File Name.

Example:

```
~ $ Ls -l
-Rw-r - r - 1 robert robert 2007-07-10 20:04 24,264,704 ReactOS.img
-Rw-r - r - 1 robert robert 2007-03-09 14:30 25,579,520 ReactOS.iso
```

The option-h displays the command, the size of the files in kilobytes, megabytes, gigabytes or terabytes of ls.

```
~ $ Ls -lh
-Rw-r - r - 1 robert robert 24M 2007-07-10 20:04 ReactOS.img
-Rw-r - r - 1 robert robert 25M 2007-03-09 14:30 ReactOS.iso
```

Both Microsoft Windows and Unix / Linux can use wildcards to filter the file name. Wildcards are special characters asterisk (*) and question mark (?). The asterisk represents any number of any characters in the filename. A question mark is on the other hand for exactly one character. The following example lists all files under Microsoft Windows with the file extension. On img.

```
C: \> dir *. img
```

The following example lists in Unix / Linux files to all the three character file names are in, the first is a character.

```
~ $ Ls a?
```

Directories created to Unix / Linux and Microsoft Windows with the command mkdir followed by the name of the new directory. On Microsoft Windows, there is the short form md.

Example on Unix / Linux:

```
~ $ Mkdir directory
```

Example, under Microsoft Windows:

```
C: \> mkdir directory
```

To delete empty directories is Unix / Linux and Microsoft Windows under the command rmdir.

```
C: \> rmdir directory
```

On Unix / Linux can whole directory trees are cleared. This command is used to rm-rf followed by the directory name. The -r option causes the deletion of all subdirectories and files it contains. The -f option will delete without asking. This command should be applied only after careful consideration. He really deletes everything embedded in subdirectories disks. On Microsoft Windows you delete directories that are not empty, rmdir / s.

```
C: \> rmdir / s list
```

Deleting files is done on Unix / Linux with the rm command, followed by the file name.

```
~ $ Rm file
```

On Microsoft Windows, files with the del command deleted.

```
C: \> del file
```

Renaming of directories and files is done on Unix / Linux with Microsoft Windows with mv and move. In each case, the old and then given the new name. Example, under Microsoft Windows:

```
C: \> move oldname newname
```

Example on Unix / Linux:

```
~ $ Mv oldname newname
```

Continue to serve the commands `mv move` and also to move files or directories in other directories. Example, under Microsoft Windows:

```
C: \> move old directory name
```

Example on Unix / Linux:

```
~ \$ Mv oldname Directory
```

Copied files under Unix / Linux with `cp`. Directories will be copied content with, the `option-r` to specify.

```
~ \$ Cp-r directory directory copy
```

On Microsoft Windows command files with the copied `copy`.

```
C: \> copy file file copy
```

Complete lists are Microsoft Windows with the command `xcopy / e` copies under.

```
C: \> xcopy / e directory directory copy
```

Links

On Unix and Linux, it is possible to reference files and directories with links to several times the file system. A distinction between hard and symbolic links. Hard links can only be applied to each data and use the same inode of the destination file. An inode number uniquely identifies a single file. This means that a hard link still works if the target file has been moved. A symbolic link, and symlink or soft link called, is merely a reference and not a real element. If one moves the target file, and a soft link points to it, the destination file does not know that a link to it, so the symbolic link points nowhere. The advantage of work that these types of links can work partitions and file system-wide and for directories. Unix / Linux is under the command `ln-s` creates a symbolic link. The following example is a symbolic link to `/usr/src/linux` to the `/usr/src/linux-source-2.6.33` created with. That is, under the path `/usr/src/linux` is the directory `/usr/src/linux-source-2.6.33` mapped the.

```
~ # Ln-s / usr/src/linux-source-2.6.33 / usr / src / linux
```

The `option-f`, an existing symbolic link is overwritten. After running the following example shows the symbolic link `/usr/src/linux` to the `/usr/src/linux-source-2.6.36`.

```
~ # Ln-sf / usr/src/linux-source-2.6.36 / usr / src / linux
```

In Microsoft Windows versions, there are links that can not be compared with a link on Unix / Linux. An installed Cygwin environment, it also allows running Microsoft Windows, when using the NTFS file system, symbolic links and hard to put up.

determine file type

In DOS and Microsoft Windows, the file type by the three characters after the dot in the file name is determined. For example, `features.exe`. Often, the display disappears from these three characters, which is a security risk. On Unix / Linux, the file name indicates not necessarily the file type. The command `file` with the file type can be determined. After the command `file`, one or more files or directories are specified. Directories on Unix / Linux and files. The following example determines the file type of the file `/etc/fstab`.

```
~ \$ File / etc / fstab
/Etc / fstab: ASCII text
```

The file `/etc/fstab` is a text file.

Text Files

On Microsoft Windows can be text files with the command's `type`.

```
C: \> type text file
```

On Unix / Linux is the `cat` command to output text files. Currently held information about the system are many Unix variants under the `/proc` text form in. The following command displays such information to the CPUs.

```
~ \$ Cat / proc / cpufreqinfo
```

On Unix / Linux text files can be comfortable with the commands show `less`. Is `less` data to the terminal size of cut, with various buttons to scroll with leaves. With `q` is `less` complete.

```
~ \$ Less text file
```

For system-error analysis or the analysis of log files is necessary. On Unix / Linux this is usually `/var/log` as text files written in. Last entries to a log file to display can be continuously used the `tail` command. `The-f` causes that even the entries that match this command in the log file be written to call. It is often used on Linux, the following command to monitor the system log file.

```
~ \$ Tail-f / var / log / messages
```

Line with certain patterns is the pipe) character (`|`) For filtering and the command `grep`. A pipe causes the redirection of the output of a command to the input of another command. That said, here are all new entries in the file `/var/log/messages` in the command passed `grep`. `Grep` can only lines by, in which the specified pattern occurs. Lines are shown here, the patterns contain the `error`, with `case-sensitive-i` option ignored by. In other words, if the system has an error is detected, it is displayed.

```
~ \$ Tail-f / var / log / messages | grep-i "error"
```

Editing text files

Under Microsoft DOS / Windows, the editor `edit` to edit files. Is called program with the `edit` command followed by the name of this text file. With the mouse or pressing the [Alt] Achieving the menus.

```
C: \> edit text file
```

On Unix / Linux powerful text editors are available. This is because all the configuration files under Unix / Linux text files. Therefore, knowledge of the operation of a ubiquitous text editor are inevitable. The powerful editor is `vi` every unix-like system exists on. Its operation is considered unusual. One reason for its strange use, `vi` is that the terminals are all to be operated. Not all terminals will have all cursor and special keys. The launch of `vi` by entering the command `vi` and the file name of the edited file as an option.

```
~ \$ Vi text file
```

The bottom line is used to display information and communicate with the user. `Vi` has three modes of operation - a factor that contributes to the fact that this editor is considered difficult to use. After the start is the `vi` command mode. Here, the cursor using the cursor keys to move anywhere. By pressing [Esc] and [i] will lead to the text input mode.

```
[Esc] [i]
```

This makes it possible to enter text. Then [Esc] must be pressed to exit the entry mode. To save you press the following keys in succession.

```
[Esc] [:] [w]
```

The editor `vi` can be the following command ends with.

```
[Esc] [:] [q]
```

If you have made a mistake, you should quit the `vi` editor without saving. Serves the following key sequence:

```
[Esc] [:] [q] [!]
```

A big advantage of *vi* is that multiple commands simultaneously the Alt, Ctrl or other modifier keys may be discontinued without pressing in succession. For experienced users this means a considerable increase in speed.

add files to each other

The command and *copy* / *b* are bound together in several Microsoft Windows files. The following example adds the files *raw-hd1.img* and *raw-hd2.img* the file *raw-hd3.img*.

```
C: \> copy / b + hd1.img raw-raw-raw-hd2.img hd3.img
```

On Unix / Linux is the joining of files using the command *cat* possible. *cat* is used to output file content. In this example, the content of *raw-hd1.img* and not *hd2.img* raw output, but one by one in the file *raw-hd3.img* redirected. Diversions are defined with the sign >. The principle of Ein-/Ausgabeumlenkung is also found in Microsoft Windows and DOS.

```
~ $ Cat raw hd1.img hd2.img raw-> raw-hd3.img
```

Compression and Archiving

As a compression method is called to reduce the memory requirement of data. The data volume is reduced by a more favorable representation is determined. This is called a lossless compression if the coded data correspond exactly to the application of the corresponding Dekodervorschrift those of the original.

ZIP

The ZIP file format is an open format for compressed archive files. The archive files usually carry the extension. *Zip*. In ZIP format, the files are compressed individually. It is also possible to distribute the archive to multiple files or to create self-extracting files. Not all compression program, whose name is the string "ZIP" contains works with the ZIP file format. On Microsoft Windows format, the ZIP the program 7-Zip support (see below). On Unix / Linux command enables the compression of the *zip* in ZIP format. The following example *MeinArchiv.zip* is a ZIP archive with the name applied to all files on the file extension. *Img* compressed with. The file name of the archive is automatically with the extension. *Zip* expanded.

```
~ My $ zip archive *. img
```

Unzip this archive with the command *unzip*.

```
~ $ Unzip MeinArchiv.zip
```

GZIP

The compression program *gzip* is available for almost all operating systems, offers a good degree of compression and is free of patented algorithms. The usual file extension for gzip-compressed files. *Gz*. Since *gzip* only compresses individual files, files are collections of several usually first with *tar* (see below) before they are compressed with *gzip* summarized. Such archives have the double extension. *Tar.gz* or *simple*. *Tgz*. On Unix / Linux is the *gzip* compression with the standard today, because many of the tasks a good compromise between high speed and good data reduction allows for. On Unix / Linux compressed file *Platte.img* the following example.

```
~ $ Gzip Platte.img
```

, Uncompress with *gunzip* takes place.

```
~ $ Gunzip Platte.img.gz
```

Alternatively, the decompression with *gzip-d* possible.

```
~ $ Gzip-d Platte.img.gz
```

bzip2

The free program *bzip2* compression is used for lossless compression of files. The *bzip2* compression is often more efficient but slower than the compression with *gzip*. With *bzip2* compressed files by file extension. *Bz2* characterized. tar files compressed with *bzip2* were typically have the extension. *tar.bz2*, *tbz* or. *tbz2*. In this example, the file is compressed *Platte.img*.

```
~ $ Bzip2 Platte.img
```

The decompression is done with *bzip2-d*.

```
~ $ Bzip2-d Platte.img.bz2
```

tar

tar is the name of the Unix business common archiving utility. The name was changed from *tape archiver* (tape archiver) which, since the program was originally backed up data to tape drives. *Tar* offers the possibility of a single sequential file to write to files, or restore files out of that. The resulting file has the extension. *Tar* and is used as a tarball (Teerklumpen) also referred to. On Microsoft Windows, tarballs are processed with 7-Zip (see below). Many popular archive programs can unpack tar archives, at least. Often, tar files compressed with *gzip* or *bzip2*. A compressed tarball is usually from the extension. *Tar.gz*, *Tgz*, *Tar.Z*, *Tar.bz2*, *Tbz2* or. *Tbz*. On Unix / Linux is the command an archive with the contents of the / etc created with. The *c* option causes the creation of the archive, *v* is used to display the archived files and *f* defined with the following option to the file name of the archive. *wil* end the directory is specified, the one archive.

```
~ $ Tar cvf etc.tar / etc /
```

Subsequently, this archive with *gzip* compressed.

```
~ $ Gzip etc.tar
```

Newer versions of *tar* support compression using *gzip* (-z option). That is, the two previous commands are executed with a command:

```
~ $ Tar czvf etc.tar.gz / etc /
```

Newer versions of *tar* support compressing with *bzip2* (option *j*).

```
~ $ Tar cjvf etc.tar.bz2 / etc /
```

For unpacking tar archives option is to *x* is necessary.

```
~ $ Tar xvf etc.tar
```

If a *gzip* compressed tar archive before, this is to decompress before.

```
~ $ Gunzip etc.tar.gz
```

Newer versions of *tar* support the decompression with *gzip* (*z* option).

```
~ $ Tar xzvf etc.tar.gz
```

Newer versions of *tar* support the Decompress with *bzip2* (option *j*).

```
~ $ Tar xjvf test.tar.bz2
```

The *option-C* enables the unpacking of the archive in another directory.

```
~ # Tar xzvf qemu-0.9.1-i386.tar.gz-C /
```

To view the contents of a tar archive that is used to select *t*.

```
~ $ Tar tvf etc.tar
```

The *tar* program does not support effective management of sparse files. That is, sparse files are filling with *tar* increases the.

7-Zip

7-Zip is a free compression utility for Microsoft Windows. With 7-Zip can be compressed (archive) files *7z* (own new format), *zip*, *gzip*, *bzip2*, and *tar* to create. The graphical user interface is part thanks to drag & drop and context menus seamlessly in Microsoft Windows. Invites you to install the installation file from the site <http://www.7-zip.org> down and start it. It is automatically installed on the target folder (*C:\ Program Files \ 7-Zip*) in demand. After installation, the *7-Zip File Manager* from the *Start menu*, *programs* are started. By clicking the right mouse button on one or more files or directories that you add the item to a 7-Zip archive add on. In the following dialog will be prompted for the type of compression and file name of the archive. To unpack an archive with a click right mouse button is to open this archive from the context menu. In Item 7-Zip to unpack the archive options are available. 7-Zip also fits in the File Explorer in Microsoft Windows. It is possible to use it have the same functions with the right mouse button. are 7-Zip it as a command line tool for Linux. The installation is done in Debian and Ubuntu using a command line.

```
~ $ Sudo apt-get install p7zip
```

A 7Zip archive is unpacked as follows.

```
~ $ P7zip-d archiv.7z
```

RAR

RAR is an algorithm and file format for compressing files. Several files are compressed together. This will also eliminate redundancies between the files. RAR supports encryption of the compressed data. The ending of this archive file is *usually*. *Rar*. Since the Entpackroutinen are freely available, the decompressing RAR archives is now supported by virtually every multi-format archiver. The compression algorithm is not released. Officially, therefore, only support from the manufacturer RAR (<http://www.rarlab.com>) published WinRAR, RAR RAR for DOS and Linux for this functionality. These tools from the site <http://www.winrar.de> downloaded and installed. On Debian and Ubuntu installation is done with a command line.

```
Host ~ $ sudo apt-get install rar
```

On Unix / Linux file decompresses the following example, the *archiv.rar*.

```
Host ~ $ rar e archiv.rar
```

Users and rights management

Unix / Linux

Unix was the beginning of a multi-user operating system. This means that multiple users can simultaneously work on the computer. It is possible to allow other users or prohibit access to certain files. Users are assigned to groups, which collectively specific access rights are granted. A special role played by the *root* user (System Administrator), the single user full access to the system as. Users typically has its own home directory, in which only he and the *root* user can create and delete files and Everyone. On Unix file system a file has rights, which are spread in three ways:

- Ownership
- Group
- Other

Each of the three user classes may have one or more of the following rights:

- *Read*: this right is shown the letter *r* by.
- *Write*: The user can write to the file or subdirectory in the directory and create files, edit, delete, rename and change their file permissions. This right is represented by the letters *w*.
- *Run*: For a file, users must run the program the file as. If a list of the bit to run, may change the user in the directory and get there files or directories. This right is represented by the letter *x*.

The rights of the three user classes are one after the other, the right of each class by a character triplet is represented. In place of the first character in the triplet is an *r* when reading bit is set. At the second site is a *w*, if writing is allowed. At the third point is *x* when running is allowed. The command *ls -l* can the rights of files to list. In this example, *messages* are the rights to the file */ var / log / display*.

```
~ $ ls -l / var / log / messages
-rw-r----- 1 root adm 44 618 2007-07-18 16:07 / var / log / messages
```

The file is owned by *root* and group *adm*. The *root* user can read and write the file (*rw*). *Adm* file, the group may only read this (*r*). All other people can read this file or modify. On Unix / Linux with the command *chmod* changes the file permissions. These changes are only on the owner of the file or by the users to perform *root*. In the following example, the file *Platte.img* for all to read but only the owner, *Robert* here, may change this file.

```
~ $ ls -l Platte.img
-rw-r--r-- 1 robert users 2007-07-10 20:04 24,264,704 Platte.img
```

The option *-g + w* is the command *chmod* also the group (*g*), where *users*, right to write (+ *w*) permission.

```
~ $ chmod g + w Platte.img
```

These rights can command *ls -l* are listed with the.

```
~ $ ls -l Platte.img
-rw-rw-r-- 1 robert users 2007-07-10 20:04 24,264,704 Platte.img
```

To all those who lack group of *users* are in the read access (*r*) to withdraw the following command is used.

```
~ $ chmod o+r Platte.img ~ $ ls -l rw-rw-rw---- 1 robert users Platte.img 2007-07-10 20:04 24,264,704 Platte.img
```

It can be right with *u* for the owner, with *g* the group and with *o* for all other users to adjust. These options can also be combined. The following command assigns the rights to read and write to the user and group.

```
~ $ chmod ug + rw Platte.img
```

Whether a file on Unix / Linux executable is not dependent on their file type. They are the respective execution rights for user, group and other necessities. The execution rights are each characterized by an *x*. For example, to make a self-written shell script executable for all the command applies.

```
~ $ chmod + x MeinSkript.sh
```

Widespread is the notation of the file permissions in octal notation. In each case represents an octal number a user class, in the order the owner, group, others. A point is in this case the sum of the three rights together as: 1 for Run, 2 for 4 for writing and reading. In order for writing and reading to award the rights, is the 6 (2 + 4) apply. The following example returns the user to the group and all others read and write privileges.

```
~ $ chmod 666 Platte.img
```

Unix / Linux are under the command *chown* the owner and group memberships change. To give a file the following command is used.

```
~ # Chown andrew: family Platte.img
```

The file is owned by *Andrea* and the user group to the *family*. In order to complete directory trees with subdirectories to change the ownership, used the *option-R*.

```
~ # Chown-R robert / home / testus
```

The *su* command is used on Unix / Linux to the user ID to change. The syntax is as follows.

```
~ $ Su - username
```

If `su - username` is called without a password, it prompts the user for a password to change to root. The dash after the `su` command causes a complete login process to be carried out. That is, it will read the complete profile of the new user and set the environment variables again. With the `exit` command, it returns to the original user and his rights back.

```
~ $ Su -
Password: *****
~ # Exit
~ $
```

`sudo` is a command on Unix / Linux, which allows the user to run processes with the rights of another user, such as the user `root` to start processes. To use `sudo` to run a `sudo` command is followed by the command input.

```
~ $ Sudo chown-R robert / home / testus
Password: *****
```

It can use the `option-p` of the text for the password request to be changed.

```
~ $ Sudo "p password" enter: chown-R robert / home / testus
Please enter your password: *****
```

Mac OS X and Ubuntu are under the root account is disabled by default, instead of the command `sudo su` is recommended.

Microsoft Windows

From Microsoft Windows 2000 programs enable the `runas` command with the privileges of another user to execute. Here, the password of another user is required. It is also possible in Windows Explorer specific file types (exe.. Msc) with the privileges of another user to start. This is possible by pressing the Shift key and right mouse button. From the context menu, select `Run as ...`.

Storage

dd, dd_rescue and dcfldd

The classic UNIX tool `dd` copies data flows between the block device files and / or regular files. It can copy entire partitions, hard disks, floppy disks and CD / DVD byte for byte. Only the existing files will not play, but it will generate an accurate picture. Unix-/Linux-Versionen `dd` is almost all sold on and there are versions for Microsoft Windows. It can be applied `dd` under Cygwin. It is possible to `dd` as a separate tool in Microsoft Windows to install. It is the URL <http://www.chrysocome.net/downloads/dd-0.5.zip> download, unzip and run directly. If the option describes the file to read. The option of describing the write file. The following example imports an inserted under Linux but not embedded disk.

```
~ # Dd if = / dev/fd0 of = floppy.img
```

Subsequently, after a disk change that image will be copied to another disk.

```
~ # Dd if = floppy.img of = / dev/fd0
```

A data stream of zeros with a defined length will be redirected to a file, creating a raw image. The following command creates an image in raw format with a size of one GB. It will be out of the device `/dev/zero` zeros got there and the file written in `Platte.img`. The block size is set at 1024 KB. The count option specifies how many of these blocks, it should be written. To generate a sparse file option, the required `seek`, the end of the file determines the. In between, no blocks are written. In the following example, a written block (`count = 1`) and file the end of 1024 set the blocks.

```
~ $ Dd if = / dev / zero of = bs = 1024k count Platte.img = 1 seek = 1024
1 +0 records in
1 +0 records out
1048576 bytes transferred in 0.002445 seconds (428871465 bytes / sec)
```

`dcfldd` (<http://dcfldd.sourceforge.net>) is an enhanced version of `dd`. Designed to be there by the U.S. Department of Defense Computer Forensics Lab. It provides additional features for computer forensics and security:

- During the transmission of the data can hash functions ensure data integrity.
- It will be displayed continuously updated information on the amount of data transferred and the date of completion.
- `dcfldd` supports secure deletion of volumes. Here, a predetermined pattern can be used.
- `dcfldd` can verify that a disk is a bit exact copy of the specified input file or pattern.
- The output can take place simultaneously on multiple files or disks.
- The output can be split into several files.
- It can display graphical AIR (<http://air-imager.sourceforge.net> used).

The installation is done under Ubuntu using a command line.

```
~ $ Sudo apt-get install dcfldd
```

The following command reads 5 GB from `/dev/sda` and writes the output to the files `sda.raw.aa`, `sda.raw.ab`, `sda.raw.ac`, `sda.raw.ad` and `sda.raw.ae`. Here are the MD5 and SHA256 hashes calculated and in the files and written `md5.txt sha256.txt`. The block size is set to 512 bytes. For read errors zeros are written.

```
~ $ Dcfldd if = / dev / sda hash = md5, sha256 hashwindow = 5G md5log md5.txt = \ = sha256.txt sha256log hashconv after bs = 512 conv = NOERROR, sync
```

With `dcfldd` it is possible to create two images simultaneously.

```
~ $ Dcfldd if = / dev / sda of = of = image1.img image2.img
```

Defective storage media fails at `dd` and it is `dd_rescue` apply. The tool is specifically developed for data recovery. `dd_rescue` has two advantages over `dd`. First, breaking the copy `dd_rescue` for read errors does not, but writes instead of the unreadable blocks an equal area zeros in the target file. This is obtained at the end of a complete image of the partition in which only the unreadable sectors are replaced by zeros. For other works `dd_rescue` with two block sizes for reads: If no error occurs, read large blocks (default: 16384 bytes). After an error, it goes further with smaller blocks (default: 512 bytes). Both values can be changed using options. Unlike `dd` can not `dd_rescue` standard output and write to a pipe. Installed, this tool available as a package `ddrescue` or it invites from the site <http://www.garloff.de/kurt/linux/ddrescue/> down. For example, my installation on Ubuntu.

```
~ $ Sudo apt-get install ddrescue
```

When calling `dd_rescue` are the typical options `dd if =` and `of =` not necessary. The first and second option specify the source and destination. While the program works, it shows how far it has progressed. For read errors, a warning is issued. In this example, a floppy disk `dd_rescue` imported with.

```
~ $ Dd_rescue / dev/fd0 bla.img
```

Partitioning

In simplified terms, partitioning the division of a physical disk into several logical areas. Many operating systems expect at least one partition per disk. Unix derivatives need at least one partition for the root directory (root). An additional swap partition is recommended. More partitions increase security and simplify administration. BSD operating systems, the division of the hard drive is called slices. Since the use of partitions or slices differently on various operating systems, the programs differ from the partitioning between the operating systems. Image files do not have to be partitioned. You mentioned in this case only one partition. Many partitioning programs are called `fdisk`. The `fdisk` version of Linux is dialogue-based and offers many options. To view all available partitions is the `fdisk -l` command.

```
~ # Fdisk -l
```

In addition, the option `u` specifies. The size of the partitions not in cylinders, but in sectors is specified.

```
~ # Fdisk -lu
```

It should be noted that all data is lost due to disk partitioning! To partition a disk *fdisk* is the name of the volume called followed. If the disc is a real hard disk, the device name to be indicated. Linux is under */dev/hda* is the first IDE drive specifically with. */Dev/hdb* marks the second IDE drive. SCSI disks on Linux with */dev/sda*, */dev/sdb*, and further symbolizes Sun In this example, the second IDE hard disk (*/dev/hdb*) to partition.

```
~ # Fdisk / dev/hdb
```

If a virtual disk is partitioned, the name of the image file must be indicated. The *option-C* is necessary for *fdisk* to specify a number of cylinders. Although the number of cylinders for a disk image does not matter, *fdisk* requires this information.

```
~ # Fdisk-C 130 Platte.img
```

The lists of commands to *fdisk m. N* with a new partition is created.

```
Command (m for help): n
Command action
  e extended
  p primary partition (1-4) p
```

From four partitions must be created an expanded (extended) partition. Here are just a partition is created. It is *p* for a primary partition must be indicated. The number of the partition is 1

```
Partition number (1-4): 1
```

The partition size is specified by the first and last cylinder. A partition, the default values are accepted.

```
First cylinder (1-130, default 1): 1
Last cylinder or + size or + or + size m sizeK (1-130, default 130): 130
Using default value 130
```

To check can you look at the partition table *p*'s with.

```
Command (m for help): p
Disk Platte.img: 0 MB, 0 bytes
255 heads, 63 sectors / track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
   Device Boot Start End Blocks Id System
Platte.img1 1130 1044193 + 83 Linux
```

The system type of partition is set by default on Linux. To change, for example, the system type to FAT32, the following steps are necessary.

```
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): L
```

The command *L* is allowed to list all system types. FAT32 is *c* selectable.

```
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

As a control we can again show the partition table *p* with.

```
Command (m for help): p
Disk Platte.img: 0 MB, 0 bytes
255 heads, 63 sectors / track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
   Device Boot Start End Blocks Id System
Platte.img1 1130 1044193 + c W95 FAT32 (LBA)
```

W write changes to disk and exit.

```
Command (m for help): w
```

Partitioning is comfortable from the **Gnome Partition editor** supports. The Gnome Partition Editor (GParted Live-CD) is not a command line tool, but a complete Linux distribution, the program includes Gnome Partition Editor. This one is independent of the partitioning programs of the individual operating systems. GParted is available as a live CD and Live USB. For the live-CD to Load the ISO file of the live CD from the URL <http://gparted.sourceforge.net/index.php> down and boot the system. It can create partitions and file systems located therein, enlarged, reduced, moved, copied and reviewed. This one creates such as space for other operating systems, reduced by the previously existing partitions. Furthermore, it is possible to rearrange partitions within a hard drive or move to other hard disks.

After starting the live CD is the default boot menu choice *GParted-liveCD (auto-configuration)* to confirm with Return. Then ask for the keyboard layout. For a German layout is entering *10th* The following languages are also entered for the 10 German. After booting seems the Gnome Partition Editor and the partitions and the free areas show the hard disks. In Gnome Partition Editor is chosen in the pull-down list (right above), the hard disk (*/dev/hda*) from. Then you click on the icon *New* to create a new partition. If only one partition to be created, the default values for the size to take it easy. In this example should be used as a FAT32 file system. In Gnome Partition Editor, click on the icon *Apply* to the hard disk to write the changes. To finish the GParted live CD with a click on the icon *Exit*.

Mounting File Systems

When mounting or attaching a file system the operating system at a certain point (mount point) provided. Usually the integration is done automatically at boot. Removable media are often automatically included in access by an automounter. Manually mount may be necessary to load a data CD or floppy disk or connect a USB stick. Only after the data on the mount point is available. In Microsoft Windows, this is another drive letter. For Mac OS X are removable by the automounter / Volumes involved in and appear on the desktop. In Linux is often a subdirectory of */mnt* or */media* involved in.

The opposite of mounting is umount. This is done by ensuring that no files are used by the file system and that all data written to the file system. If a file system without taking away, for example, by removing a USB stick, it can cause data loss. On Unix / Linux, the second partition on the primary hard disk on the first ATA port is mounted manually using the command:

```
~ # Mount / dev/hda2 / my / directory
```

An umount command is *umount with*. Here is an example of the indication of the device.

```
~ # Umount / dev/hda2
```

Here is an example of specifying the mount point.

```
~ # Umount / my / directory
```

A list of mounted file systems is the *mount* command without options:

```
~ $ Mount
```

With the *-t* option, the type of file system are specified. In order to access an NTFS image, for example, used the *option-t ntfs*.

```
~ # Mount-t ntfs ntfs.img / mnt
```

Also, a file can be on the loop device in turn incorporated as a file system, most often this is found in ISO images from CD / DVDs using:

```
~ # Mount-r-t iso9660-o loop DVD.iso / my / directory
```

The assignment of the file systems to the device name may change in the operating system. Why a file system is cheaper to integrate with its unique UUID (Universally Unique Identifier). First is the command `blkid` the UUID to be determined.

```
blkid ~ # / dev/hda2
/ Dev/hda2: UUID = "571bba46-083f-45a4-abc6-f15822718453" SEC_TYPE = "ext2" TYPE = "ext3"
```

The mount command is used as follows.

```
~ # Mount UUID = 571bba46-083f-45a4-abc6-f15822718453 / my / directory
```

partitions determine the level

On Unix / Linux command the level of the involved partitions with the calculated `df`. The `-h` used for the size information kilobytes, megabytes and gigabytes. This serves to improve readability.

```
$ df-h
Filesystem Size Used Avail Use% Mounted on
/ Dev/hda3 54G 46G 15% 8.0 g /
/ Dev/hd1 111M 69M 36M 66% / boot
```

Network configuration and test

Today's networks are mostly based on the TCP / IP. The acronym TCP / IP stands for Transmission Control Protocol (TCP) and Internet Protocol (IP). TCP / IP is mature and has excellent routing capabilities. These logs are the basis for the Internet. Their application protocols such as HTTP, FTP and SSH, use TCP / IP requires. In addition to TCP / IP, there are other network protocols such as NetBIOS. NetBIOS is still used occasionally in the DOS / Windows world. On Unix / Linux is used for manual network configuration, the `ifconfig` program. Unfortunately, the options for the Unix / Linux derivatives are not uniform. To display network interfaces to be-all, you call `ifconfig` with no options.

```
~ # Ifconfig
```

In some versions of Unix, the `-a` option to attach.

```
~ # Ifconfig-a
```

If information is listed only one network interface, the name of the network interface to be indicated. In this example, information is the network interface `eth0` to appear. The most important information, this interface is that it even exists and what IP address it has.

```
~ # Ifconfig eth0 eth0 Link encap: Ethernet HWaddr 00:0 B: 6B: 69: E8: 63 inet addr: 192.168.1.10 Bcast: 192.168.1.255 Mask: 255.255.255.0 inet6 addr:
```

To an existing network interface an IP address, for example 192.168.1.15, assigned to the command in Linux is required.

```
~ # Ifconfig eth0 192.168.1.15 netmask 255.255.255.0 up
```

The `eth0` option marks the first network card in the computer. Next the IP address is specified. This IP address may not be awarded on the net. The option `netmask` defines the network mask. This option adds `up` that the network interface is enabled. To interface to disable a network option is the use `down`.

```
~ # Ifconfig eth0 down
```

On Microsoft Windows, the command / `all` information on network interfaces on `ipconfig`.

```
C: \> ipconfig / all
```

With `ipconfig / renew` network adapter, the IP addresses of all renewed. This only makes sense if the network configuration via DHCP is. With `ipconfig / release` to be released all the compounds.

The command `ping` is checked whether a particular network interface in a network is reachable. `Ping` is under many operating systems. As an option, the DNS name or IP address of the tested network interface is specified. This example is Linux, the connection to the host `qemu-buch.de` tested with `ping`.

```
~ $ Ping-buch.de qemu
PING qemu-buch.de (85.214.74.183) 56 (84) bytes of data.
64 bytes from tr-e.de (85.214.74.183): icmp_seq = 1 ttl = 53 time = 9.0 ms
64 bytes from tr-e.de (85.214.74.183): icmp_seq = 2 ttl = 53 time = 1.6 ms
```

The answers can be seen that the host is reachable. If the other side does not respond, this does not mean she is not available. The `ping` command uses Internet Control Message Protocol (ICMP). It is used in networks to exchange information and error messages via the Internet Protocol (IP). It is expected of every router and every computer to support ICMP. There are computers masquerading and firewalls, which can not by this protocol. For example, blocks the QEMU user mode network stack of the protocol. Ie `ping` does not work in a QEMU instance. We tried in this case, a `telnet www.qemu-buch.de 80` After the prompt you type in the web server / `index.php` GET and get a Web page in raw format.

Network services and clients

SSH

Secure Shell (ssh) ist sowohl ein Programm als auch ein Netzwerk-Protokoll, mit dessen Hilfe man sich über eine verschlüsselte Netzwerkverbindung auf einem entfernten Computer einloggen kann. SSH ermöglicht eine sichere, authentifizierte und verschlüsselte Verbindung zwischen zwei Rechnern. Mit OpenSSH ist auch eine freie Implementierung von SSH vorhanden, die mittlerweile einen großen Verbreitungsgrad erreicht hat. SSH-Implementierungen waren ursprünglich nur unter Unix verfügbar, mittlerweile wurden jedoch sowohl SSH-Server als auch Clients für andere Plattformen programmiert. Unter Cygwin gibt es einen SSH-Server für Microsoft Windows, der auf OpenSSH basiert. Dieser ermöglicht das Einloggen per SSH auf einer Microsoft Windows-Maschine. Ein weiter SSH-Server für Microsoft Windows ist freeSSHd (<http://www.freesshd.com>). Um sich zum Beispiel als Systemadministrator (root) auf den Rechner mit dem Namen `bibo` mit `ssh` einzuloggen, dient folgender Befehl.

```
~$ ssh root@bibo
```

Die Option `-X` ermöglicht das X-Forwarding. Dabei werden die grafischen Ausgaben des Rechners, auf den man sich eingeloggt hat, auf den eigenen Rechner umgeleitet.

```
~$ ssh -X root@bibo
```

Normalerweise lauscht ein SSH-Server auf dem Port 22. Dies kann verändert werden. Um sich mit dem SSH-Client auf einem anderen Port zu verbinden, ist die Option `-p` gefolgt von der Port-Nummer anzugeben.

```
~$ ssh -p 12345 root@bibo
```

putty

Download: <http://www.putty.nl/download.html>

Für Microsoft Windows steht mit `putty` ein SSH-Client zur Verfügung. Dazu lädt man die Datei `putty.exe` herunter und startet diese. Um sich zum Beispiel als Systemadministrator (root) auf dem Rechner mit dem Namen `bibo` mit SSH einzuloggen, ist folgende Konfiguration einsetzbar. Möglicherweise muss in der Einstellung der Windows-Firewall auf dem Host dieser Port freigegeben werden.

```
Host Name: bibo
Port: 22
Protocol: ssh
```

SCP

Secure Copy (SCP) ist ein Protokoll sowie ein Programm zur verschlüsselten Übertragung von Daten zwischen zwei Computern über ein Netzwerk. Es gewährleistet Vertraulichkeit, Integrität und Authentizität der übertragenen Daten. Dazu nutzt es das Protokoll SSH. Mit OpenSSH steht ein freier Server sowie ein freier Client für SCP zur Verfügung. Das Programm `scp` funktioniert in Grundzügen wie der Kopie-Befehl `cp`. Im folgenden Beispiel werden von dem Rechner mit dem Namen `bibo` die Datei `/etc/passwd` in das aktuelle Verzeichnis kopiert.

```
~ $ scp root@bibo:/etc/passwd .
```

In diesem Beispiel wird die Datei *Platte.img* auf den entfernten Rechner mit dem Namen *bibo* in das Verzeichnis */tmp* kopiert.

```
~ $ scp Platte.img root@bibo:/tmp
```

WinSCP

Download: <http://winscp.net/eng/docs/lang/de>

WinSCP is a graphical SCP client for Microsoft Windows and can also be used as an FTP client. WinSCP is available under the GNU General Public License. For the German version you have additional Multi-Language Installation Package download this. The installation will start first in English. You select the installation dialog from the German. The installation of WinSCP is simple and self explanatory.

SSHFS

Website: <http://fuse.sourceforge.net/sshfs.html>

Das Secure Shell FileSystem (SSHFS) unterstützt unter Linux das Mounten eines Dateisystems eines anderen Rechners mit der Secure Shell (SSH) durch nichtprivilegierte Benutzer. Dabei wird das Filesystem in Userspace (FUSE) angewendet. Auf dem entfernten Rechner ist nur ein SSH-Server notwendig. Auf dem lokalen Rechner ist das Paket *sshfs* notwendig. Die Installation unter Debian/Ubuntu ist mit einer Befehlszeile erledigt.

```
~$ sudo apt-get install sshfs
```

Weiterhin wird ein Verzeichnis als Mountpoint benötigt.

```
~$ mkdir ~/sshfs
```

Folgender Befehl hängt das Verzeichnis */tmp* des Rechners *bibo* ein.

```
~$ sshfs root@bibo:/tmp/ ~/sshfs
```

Mit dem Befehl *fusermount* wird der Mountpoint wieder freigegeben.

```
~$ fusermount -u ~/sshfs
```

FTP

Das *File Transfer Protocol* ist ein Netzwerkprotokoll zur Dateiübertragung über TCP/IP-Netzwerke. Es wird benutzt, um Dateien vom Server zum Client (Download), vom Client zum Server (Upload) zu übertragen. Außerdem können mit FTP Verzeichnisse angelegt und ausgelesen, sowie Verzeichnisse und Dateien umbenannt oder gelöscht werden. Da FTP ein textbasiertes Protokoll ist, werden sowohl die Daten als auch die Authentifizierungsinformationen im Klartext übertragen. Zum Download wird ein FTP-Client benötigt. Jeder moderne Web-Browser enthält einen FTP-Client für den Download. Der FTP-Client *ftp* ist auf fast jedem Betriebssystem verfügbar. Der ursprünglich für Unix programmierte Client wurde schon bald auf andere Betriebssysteme, wie zum Beispiel Microsoft Windows, portiert. Das Programm *ftp* wird als Kommandozeilenbefehl gestartet, oft mit Angabe eines FTP-Servers. Bei erfolgreicher Verbindung fragt das Programm nach Benutzernamen und Passwort. In diesem Beispiel wird mit dem FTP-Client in der DOS-Eingabekonsole unter Microsoft Windows die Datei *bla.txt* zum Rechner mit der IP-Adresse 10.0.2.2 übertragen.

```
Gast C:\> ftp 10.0.2.2
Benutzer (10.0.2.2): robert
Kennwort: *****
Ftp> put bla.txt
Ftp> quit
```

Weitere Kommandos sind:

```
Ftp> ascii
```

Einstellen von ASCII als Übertragungsmodus. Dies ist der Standardübertragungsmodus.

```
Ftp> binary
```

Einstellen von Binary als Übertragungsmodus.

```
Ftp> cd Verzeichnis
```

Wechselt in das Arbeitsverzeichnis auf dem FTP-Server.

```
Ftp> delete Datei
```

Löscht die Datei auf dem FTP-Server.

```
Ftp> dir
```

Zeigt den Inhalt des aktuellen Arbeitsverzeichnisses auf dem FTP-Server an.

```
Ftp> get Datei
```

Kopiert eine Datei vom FTP-Server auf den Client.

```
Ftp> lcd Verzeichnis
```

Wechselt das Verzeichnis auf dem Client.

```
Ftp> ls
```

Lists the contents of the directory to.

```
Ftp> mget files
```

Copies multiple files from the server to the client. Wildcards are allowed.

```
Ftp> mkdir directory
```

Creates a directory on the FTP server.

```
Ftp> mput files
```

Copies multiple files from the client to the FTP server. Wildcards are allowed.

```
Ftp> pwd
```

Prints the current working directory on the FTP server.

```
Ftp> quit
```

Close connection to FTP server and exits.

```
Ftp> rmdir directory
```

Deletes a directory on the FTP server.

TFTP

The Trivial File Transfer Protocol (TFTP) is a simple file transfer protocol, and supports only the reading or writing files. Not present many features of the powerful FTP such as rights issue, the ads are existing files or user authentication. Motivation for developing TFTP was the loading of operating systems or configurations on the network. For this, the connection-oriented TCP and FTP based on it far too complex. TFTP was kept deliberately simple, however. Microsoft Windows brings house from a TFTP client (*tftp.exe*) with. Certain types of computer worms such as *W32.BlastDoor* use TFTP server for their dissemination. Therefore *tftp.exe* when using a personal firewall by default, the Internet access will be denied. A TFTP client for Microsoft Windows versions can also <http://www.tftp-server.com/tftp-download.html> WinAgents TFTP Client) download and run directly (.

```
C: \> tftp -i 10.0.2.2 GET foo.txt
```

On Unix / Linux a tftp client is often not installed. If the host system on Linux Debian derivative, is to install the package with a command line *tftp* done.

```
Host ~ # apt-get install tftp
```

Downloading a file via TFTP done with these commands.

```
Guest ~ $ tftp  
(To) 10.0.2.2  
tftp> binary  
tftp> get foo.txt  
tftp> quit
```

wget

Download: <http://www.gnu.org/software/wget/>

The tool *wget* is a free command line program to download files. The supported protocols include FTP, HTTP and HTTPS. The program is there for both Unix and GNU / Linux and Microsoft Windows. *Wget* is included in Cygwin. *Wget* can resume a broken download sites and download complete. To file from a Web or FTP server to download one, it is sufficient to *wget* the URL and access the, for example:

```
~ $ Wget http://download.savannah.gnu.org/releases/qemu/qemu-0.12.4.tar.gz
```

Telnet

Telnet (Teletype Network) is the name of the web to enable wide-spread network protocol and is used to provide users access to computers via the command line. Due to the lack of encryption it is hardly used anymore. It is also the name of a Telnet client program, which links to a remote computer can be built. The Telnet protocol uses TCP and the clients connect through port 23 to the target computer. This port can modify as with most Internet protocols. It is also possible to set up a telnet client, an interactive TCP connection to other Internet services. In the following example, the Telnet client to the host and its port is connected bibo 4444

```
~ $ Telnet bibo 4444  
Trying 127.0.0.1 ...  
Connected to localhost.  
Escape character is '^]'.
```

On Microsoft Windows menu, the Telnet client on the *Run* in the Start point distance. In the appearing command-line gives you the command *telnet*. In the menu of the client you are connecting to, the *remote system* host and port.

netcat

Download: <http://netcat.sourceforge.net/>

Das Programm *netcat*, auch *nc* genannt, ist ein einfaches Werkzeug, um Daten von der Standardeingabe oder Standardausgabe über Netzwerkverbindungen zu transportieren. Es arbeitet als Client oder Server mit den Protokollen TCP und UDP. Eine Windows-Version ist in Cygwin enthalten. Diese Version wird mit *nc* aufgerufen. In dem nachfolgenden Beispiel wird die Datei *bla.txt* von dem Rechner 1 mit *netcat* auf den Rechner 2 mit der IP-Adresse 172.16.10.61 übertragen. Dazu wird zuerst der Rechner 2 empfangsbereit geschaltet.

```
Rechner 2 ~$ netcat -w30 -vvvnp 3333 > bla.txt
```

Die Option *-w* gibt das Timeout an. Das heißt, in dem Beispiel wartet *netcat* maximal 30 Sekunden auf das Zustandekommen der Verbindung. Die Optionen *-vv* bewirken eine ausführlichere Ausgabe, um mögliche Fehler zu erkennen. Die Option *-n* schaltet *netcat* in den Lausch-Modus. Die Option *-p* bewirkt, dass keine DNS-Abfrage erforderlich ist. Es ist die IP-Adresse statt des DNS-Namens anzugeben. Die Option *-p* definiert den Port (hier 3333). Der empfangene Datenstrom wird in die Datei *bla.txt* umgeleitet. Auf Rechner 1 startet man die Übertragung der Datei *bla.txt*. Zusätzlich ist hier die Adresse des Ziel-Rechners anzugeben. Dieses Beispiel enthält keinerlei Absicherungen.

```
Rechner 1 ~$ netcat -w 30 -vvv 172.16.10.61 3333 < bla.txt
```

Viele Hacker/Cracker benutzen Netcat um Backdoors (Hintertüren) auf den Rechnern ihrer Opfer zu öffnen. Deshalb stufen einige Anti-Viren-Programme das Windows-Executable *nc.exe* als Hacking-Tool ein und verhindert dessen Ausführung.

TightVNC

Download: <http://www.tightvnc.com/>

TightVNC ist ein VNC-Server und -Client der unter der GPL-Lizenz steht. Er ist für Microsoft Windows sowie für zahlreiche Unix-Derivate verfügbar. Weiterhin gibt eine plattform-unabhängige Implementierung in Java (nur als Client). For example, my installation on a Linux Debian derivative.

```
Host ~ $ sudo apt-get install xtightvncviewer
```

Auf der Website des Projekts steht die Software für viele Betriebssysteme zum Download bereit. Zur Installation unter Microsoft Windows ist die Datei *tightvnc-1.3.9-setup.exe* herunterzuladen und aufzurufen. Im Wizard für die Installation gibt man das Ziel-Verzeichnis ein (C:\Programme\TightVNC). Danach sind die zu installierenden Software-Komponenten (Server, Viewer, Dokumentation) auszuwählen. Anschließend ist der Name des Ordners im Start-Menü festzulegen (TightVNC). Im nächsten Schritt ist anzugeben, ob der VNC-Server als System-Dienst registriert wird und ob der VNC-Server automatisch gestartet wird. Dies ist nur zu empfehlen, wenn dieser Rechner dauerhaft per VNC erreichbar sein soll. Nach der Installation wird TightVNC wie folgt aufgerufen: Menü Start, Alle Programme, TightVNC, TightVNC Viewer (Fast Compression).

Remote Desktop Protocol

The Remote Desktop Protocol (RDP) from Microsoft allows the screen of a Microsoft Windows computer on a different computer display, and in turn keyboard and mouse movements to send back. In order for a remote access is possible on the desktop of another computer. This is one of the two systems is working as a terminal server. This terminal server generates screen output on the terminal client. In addition to the expenditure of the screen and the input from the keyboard or mouse to the sound output to the terminal client will be redirected. Furthermore, the use of a printer of the terminal client is possible. RDP is based on the ITU T.128 protocol and uses TCP / IP requires. By default, the port 3389 (TCP) is used. When RDP Server Microsoft Windows Terminal Server 4.0, Microsoft Windows Server 2000, 2003 and 2008, NetMeeting, Microsoft Windows XP, Microsoft Windows Vista and Microsoft Windows uses 7th The respective Home and Starter Edition are not suitable as an RDP server. RDP clients exist for most operating systems. The virtualization software VirtualBox has its own RDP server.

As an example, the configuration of the RDP server on Windows XP. If the computer in a domain, then the following settings via Active Directory may be limited. The login must be done with a password. This is the Start Menu, Control Panel, user accounts, assign a password to an account change. On the Start menu, Control Panel, user accounts, change the way users log options are the Use the Welcome screen and use to enable Fast User Switching. Furthermore, in the control system, remote, select Allow users tab, establish a remote connection to activate. Select Remote Users ... Below are corresponding user specified. In order to speed up the login process, is in the Control Panel under Administrative Tools, Terminal Services Startup Type to Automatic to set the services. The Windows Firewall is open for port 3389. The color depth for the RDP client should be adjusted in the registry. First, to the Registry Editor is open (Start, Run, regedit32). After changing to the key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp. The value ColorDepth 4 is changed. The color depth is increased to a 24 bit.

On Microsoft Windows XP and there is a new RDP client on the Start menu, All Programs, Accessories, Remote Desktop connection. In Unix-like operating systems and OS / 2 and eComStation program is the open source *rdesktop* (<http://www.rdesktop.org>) was used. On Debian / Ubuntu package is the installation of a command line does *rdesktop*.

```
~ $ Sudo apt-get install rdesktop
```

rdesktop for example, called with the following options. The transfer is compressed with-z. The-k option sets the keyboard layout set. With the-g option, the screen resolution is set.

```
z ~ $ rdesktop-g 1000x800-k de hostname
```

Should not the entire desktop, but only to an application to be accessed on, the tool can seamlessrdp (<http://www.cendio.com/seamlessrdp/>) are used.

Synergy

Website: <http://synergy2.sourceforge.net/>

The program Synergy (open source) allows the use of a mouse and a keyboard to several computers on the network. A computer is configured as a Synergy server. On the other computers, Synergy is started as each client. When the mouse cursor exceeds the boundary of each desktop, manages Synergy further input from the mouse and keyboard over the network from the host PC to the appropriate clients. The desktops all connected computers are connected to a desktop. Furthermore, similar to Synergy from the contents of the clipboard of the computer. Since Synergy for Microsoft Windows (95/NT) and all Unix-like operating systems (Mac OS X, Linux, Solaris) is available, can the desktops of computers are connected to different operating systems. For Microsoft Windows, the installer can be downloaded from the website and run them. After installation, the configuration of Synergy by GUI is possible. On Debian / Ubuntu installation is done with a command line.

```
~ $ Sudo apt-get install synergy quicksynergy
```

For a simple configuration is to start *quicksynergy*

```
~ $ Quicksynergy
```

Should the computer as a Synergy server is working, at least one of the fields *Above*, *Left*, *Right* and *Below* the IP address of a client entered into. Should the computer be configured as a client is under the rider *use* the IP address of the server must only be shown. For detailed instructions on the Web at Synergy <http://www.elsniwiki.de/index.php/Main/SynergyKonfigurieren> .

dnsmasq

Download: <http://thekelleys.org.uk/dnsmasq/doc.html>

For small networks is the simple DHCP and DNS server *dnsmasq* recommended. DNS queries according to the file */etc/hosts* resolved or forwarded to other DNS servers. Instructions for configuration can be found at <http://wiki.ubuntuusers.de/Dnsmasq> and <http://www.martin-bock.de/pc/pc-0604.html> . The library also uses *libvirt dnsmasq* to host systems using DHCP to assign IP addresses (see http://qemu-buch.de/d/Managementtools/_libvirt-Tools).

Compiling

To software under Unix / Linux to compile from source, applied after downloading and unpacking the sources, the following steps in the created directory:

```
~ $ . / Configure
~ $ Make
~ $ Sudo make install
```

As a first step, the *script. / Configure* call. Possible options shows *. / Configure* to *help*. The *script. / Configure* checks if all the associated components are available for configuration. It also specifies conditions according to the determined configuration files and any subdirectories to *make an*. This configuration file called *Makefile* and each are in the directories created and describe what *make* is to perform there. Then the call is made of *make*. The program will *make* commands, depending on conditions, the makefiles are described in the from. It is used for example to control in a project that consists of many different files with source code that automates all the steps. Sometimes must first *make* the package to be installed. In the last step, which is running with root privileges, system with *make install* the generated files (binaries, configuration files, manuals) in the directories of your copies.

In some Distributionen the tools for compiling are not installed by default. In Ubuntu the package is *essential* to install *build*. In other distributions, *make*, *gcc*, and the packages needed.

```
~ $ Sudo apt-get install build-essential
```

Self-compiled software is not managed by the package manager. This causes the software not in the list of installed packages and can not be uninstalled so easily. Remedy the *checkinstall* program. In order to build Slackware TGZ, RPM or DEB packages when compiling. The installation is done in Debian / Ubuntu using a command line.

```
~ $ Sudo apt-get install checkinstall
```

Instead of *make install* command is used when compiling the *checkinstall*. A support can be obtained with *checkinstall - help*.

```
~ $ . / Configure
~ $ Make
~ $ Sudo checkinstall
```

The program provides *checkinstall* some questions that the specifications can be answered with mostly default. Clearly, however, the package name is to be specified. In the current directory the package is created and installed. On Debian / Ubuntu it lists the command *dpkg-l* and *dpkg-r* uninstall it if necessary.

<<< | # # | >>> <http://qemu-buch.de>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_N% C3% BCtzliche_Tools "

This page has been accessed 18,331 times. This page was last updated on 30 September 2010 at 12:04 clock changed. Content is available under GNU Free Documentation License 1.2 .

Virtualize and emulators

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators]])

<<< | # # # | >>> | English

Further virtualizer and Emulators

Here are some other virtualizer and emulators are briefly introduced.

- chroot (New root directory as a sandbox under Unix / Linux)
- Basilisk II (68k Macintosh Emulator)
- Bochs (x86 emulator)
- FAUMachine (hardware emulator, virtualization solution)
- Gxemul (ARM, MIPS, PowerPC and SuperH-Emulator)
- Linux Containers (virtualization at the OS level)
- personal alpha (alpha-Emulator)
- Solaris Zones (virtualization at the OS level)
- VirtualBox (native virtualization, full virtualization)
- Virtual PC (Native Virtualization)
- VMware ESXi (type-1 hypervisor)
- VMWare Player (desktop virtualisation)
- VMware Server (Native Virtualization, type 2 hypervisor)
- VMware Workstation (Native Virtualization)
- Win4Lin, Win4BSD, Win4Solaris
- Xen (paravirtualization and full virtualization, Type-1 hypervisor)

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren "

This page has been accessed 5138 times. This page was last updated on 27 September 2010 at 06:35
clock changed. Content is available under GNU Free Documentation License 1.2 .

chroot debootstrap, cdebootstrap, rpmstrap, jail, Debian, Ubuntu

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / chroot]])

<<< | # # # | >>> | English

debootstrap and chroot

debootstrap is used to the Linux distributions Debian and Ubuntu installation without the use of media in a chroot environment to install the tool. The installation of Debian and Ubuntu *debootstrap* is a command line does under. The package *debootstrap* is also available for other Linux distributions.

```
Host ~ $ sudo apt-get install debootstrap
```

A Ubuntu Intrepid (8.10) in the directory to install *intrepid-chroot*, for example, the following command is used.

```
Host sudo debootstrap intrepid intrepid-chroot \
      http://de.archive.ubuntu.com/ubuntu/
```

Here, a Linux directory tree is created in this subdirectory.

```
Host ~ $ ls intrepid chroot
bin boot dev etc home lib mnt lib64 media
opt proc root sbin srv sys tmp usr var
```

An old and rudimentary virtualization solution is the on almost any existing Unix-/Linux-System dar. *chroot chroot* on Unix / Linux systems allows the root directory to change. This environment will be a somewhat closed the chroot achieved. *Chroot* is a simple form of virtualization at the OS level and paves the way for Solaris Zones and FreeBSD Jails. If a chroot environment set up in a directory, for example with *debootstrap*, log into the *chroot* chroot in-one environment.

```
Host ~ $ sudo chroot. / Chroot-intrepid /
```

The *debootstrap* scale with Linux directory tree appears here in the root directory.

```
Guest ~ $ ls /
bin boot dev etc home lib mnt lib64 media
opt proc root sbin srv sys tmp usr var
```

Some commands, such as *ifconfig*, are applied in this environment is not available. This is because the / *proc* is empty. Installing packages is possible. This is useful to test a software package without having to change the host system do.

```
Host ~ # apt-get install qemu
```

With *exit* logs you off.

```
Guest ~ $ exit
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_chroot "

This page has been accessed 4828 times. This page was last updated on 27 September 2010 at 06:35 clock changed. Content is available under GNU Free Documentation License 1.2 .

BasiliskII linux osx Mac OS 7.6.1 installation iso download emulator apple macintosh

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and Emulators / Basilisk II]])

<<< | # # # | >>> | English

Basilisk II

Installation

Website: <http://basilisk.cebix.net>

Basilisk II is a 68k Macintosh emulator. This can be run by older versions of Mac OS. Basilisk II will run on many operating systems (BeOS, Linux, Solaris, FreeBSD, NetBSD, IRIX, AmigaOS 3.x, Microsoft Windows and Mac OS X). A guide to installing Mac OS X Basilisk II is under the URL <http://www.netdot.ch/~sprouting/tutorials/basilisk/>. A guide to installation on Microsoft Windows can be found at the URL <http://www.emaculation.com/articles/intro.html>. In Ubuntu the package is installed with a command `basilisk2`.

```
Host ~ # apt-get install basilisk2
```

The program is called with *Basilisk II*.

```
Host ~ $ BasiliskII
```

Mac OS 7.6.1 as a guest system

Fortunately, Apple, for the installation disks for older Mac OS versions available for download. Cheaper is to use an installation CD, as the image under <http://www.netdot.ch/~sprouting/tutorials/basilisk/> will be offered. The download of the files with `wget` or may be a web browser. The files are copied into a subdirectory. Here are the Unix commands.

```
Host $ mkdir ~ BasiliskII host ~ $ cd ~ $ wget BasiliskII host \ ~ http://www.netdot.ch/~sprouting/tutorials/basilisk/files/mac.rom.zip host ~
```

The image of the installation CD (*MacOS761.img*) is the rider with *add volumes* mounted under. Furthermore, *volumes* Create a virtual disk created under. The name is *macsystem.img* and size at least 100 MB of pretending as. The other settings, see screen shots. Start the emulator with the *Start* button. Installing Mac OS 7.6.1 is simple and self explanatory. Once installed, the image will be removed the installation CD in the Basilisk II GUI. An important tool for Mac OS 7.6.1 is the Stuffit Expander. The image with the Stuffit Expander is added in the GUI under Mac OS, the installation program called *Aladdin expand 7.6.1*.

```
Host ~ $ wget http://www.emaculation.com/articles/starterdisk.zip
Host ~ $ unzip starterdisk.zip
```

Other software can be installed where it is downloading them in the host system. Basilisk II offers a path to the host system as a mounted disk on the desktop. the downloaded software is in the path of the host system, the respective installation programs can be started. Here is a selection of software for the System 7.6.1:

```
wget http://download.system7today.com/disinfectant371.sea.hqx
wget http://www.emaculation.com/basilisk/corelwp35e.sea.bin
wget http://download.system7today.com/SimpleEdit_3.3.1.sit.hqx
wget http://download.system7today.com/vnc/VNC_68k.sit
wget http://www.emaculation.com/basilisk/iCab_Pre2.99b_English_68k.sit
wget http://www.emaculation.com/basilisk/communicator4.08_68k.bin
wget http://www.emaculation.com/basilisk/MacIRC_68K.sit.hqx
wget http://www.emaculation.com/basilisk/mpeg-dec-311.hqx
wget http://download.system7today.com/MacSSH68k.sit
wget http://download.system7today.com/movieplayer2.sit
wget http://download.system7today.com/mapping_pro_3.0.4.sit
wget http://download.system7today.com/mac-uptime.hqx
wget http://download.system7today.com/networktime.sit
```

Information about network configuration is obtained under <http://basilisk.cebix.net/README>. The configuration of the Basilisk II GUI on Unix / Linux in the file `~/.Basilisk_ii_prefs` saved. Here's an example with the above set values.

```
disk / home/hans/BasiliskII/MacOS761.img
disk / home / hans / BasiliskII / machd.img
extfs /
screen win/0/0
seriala / dev/ttys0
serialb / dev/ttys1
udptunnel false
UDPPort 6066
rom / home / hans / BasiliskII / MAC.ROM
boot drive 0
boot drive 0
ramsize 134217728
frame skip 1
modelid 14
CPU 3
fpu false
nocdrom false
nosound true
noclipconversion false
nogui false
jit false
jitffp false
JITDebug false
jiticachesize 0
jitzlazyflush false
jitininline false
type 5 keyboard
keycodes false
mouse wheel mode 1
mouse wheel lines 3
dsp / dev / dsp
mixer / dev / mixer
ignoresegv false
idlewait true
```

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_Basilisk_II "

This page has been accessed 4383 times. This page was last updated on 27 September 2010 at 06:35 clock changed. Content is available under GNU Free Documentation License 1.2 .



The BasiliskII GUI.



The BasiliskII GUI - Rider SCSI.



The BasiliskII GUI - Rider Graphics / sound.



The BasiliskII GUI - Rider Keyboard / Mouse.



The BasiliskII GUI - Rider Serial / Network.



The BasiliskII GUI - Tab Memory / Misc.



Mac OS 7.6.1 under BasiliskII.

Bochs emulator download installation windows linux qemu bximage

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / bochs]])

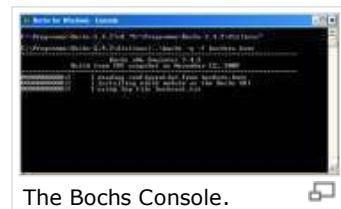
<<< | # # # | >>> | English



Bochs - Installation under Microsoft Windows.



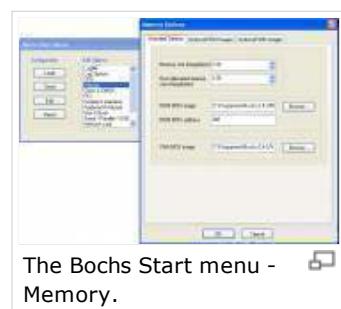
Demo Linux in Bochs.



The Bochs Console.



The Bochs Start menu.



The Bochs Start menu - Memory.

Contents

- 1 Bochs
 - 1.1 Installation
 - 1.1.1 Microsoft Windows
 - 1.1.2 Linux 1.1.2
 - 1.1.3 FreeBSD 1.1.3
 - 2.1 Creating Virtual Machines
 - 1.3 Further

Bochs

Website: <http://bochs.sourceforge.net>

Bochs is an x86 emulator (Open Source), which on many operating systems, and among other processors, running. The aim of the developers is to achieve a full PC compatibility. It is possible to install different guest operating systems under Bochs. However, the execution speed is not high. Bochs is therefore used mainly for developing and testing.

Installation

Microsoft Windows

To install on Microsoft Windows to Load installer from the site at the point *Get* down the *Bochs* and start it. A wizard helps with the installation. In step *Choose Components* to enable all components, even *DLX Linux demo*. The next step is the installation path *C: \ Program Files \ Bochs-2.4.2* be changed. After installation, check the icon *Demo Linux in Bochs 2.4.2* clicked on. It is Bochs started with a Linux guest. It logs in with *root*. A password is not set. To create a new virtual machine, with Bochs 2.4.2 is the *Start menu*, *Programs, Bochs 2.4.2, Bochs* started.

Linux

Installing Bochs on Debian and its derivatives, as Ubuntu is, as a *root* user command line done with. Otherwise sources are, as usual, *make* and *make install* to compile with *configure*.

```
Host ~ # apt-get install bochs bximage
```

FreeBSD

For FreeBSD and its derivatives, such as PC-BSD, the installation is performed using the following commands.

```
Host ~ # cd /usr/ports/emulators/bochs
Host ~ # make install clean
```

Creating virtual machines

It is convenient to apply for each virtual machine directory and save the configuration file and image files. In this example, the operating system CP/M86 be installed.

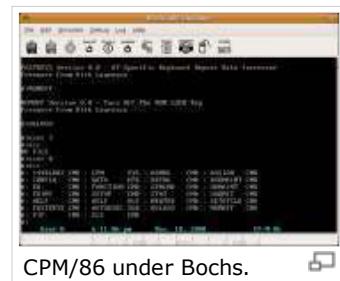
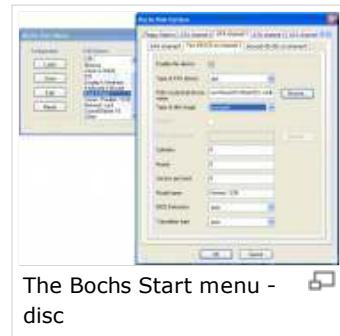
```
Host ~ $ mkdir-p ~ / bochsVMs/cpm86
Host ~ $ cd ~ / bochsVMs/cpm86
```

It downloads the image of CP/M86-Diskette and unpacked it.

```
Host ~ $ wget \
    http://www.gaby.de/ftp/pub/cpm/sysdisks/cpm86/86raw144.zip
Host ~ $ unzip 86raw144.zip
```

The tool is a virtual hard drive *bximage* created. At the end of the dialogue of *bximage*, the reference to how to access the virtual hard disk in the file *bochsrc* to configure. With a text editor to add this line.

```
Host ~ $ bximage
Do you want to create a floppy disk image or a hard disk image? Please type hd or fd. [Hd] [Enter]
What kind of image should I create? Please type flat, sparse or growing. [Flat] [Enter]
Enter the hard disk size in megabytes, between 1 and 129 023 8
What should I name the image? [C.img] [Enter]
Writing: [] Done. [Enter]
The following line should appear in your bochsrc:
ata0-master: type = disk, path = "c.img", mode = flat, cylinders = 16 heads = 16, spt = 63
```



The configuration of a virtual machine is under Bochs with a configuration file called *bochsrc*. An example of a configuration file can be used as a template.

```
Host ~ $ cp / usr / share / doc / bochs / examples / bochsrc.gz.
Host ~ $ gunzip bochsrc.gz
```

For this example, the file *bochsrc* following content.

```
config_interface: wx
display_library: wx
ROMIMAGE: file = / usr / share / bochs / BIOS-bochs-latest
cpu: count = 1, reset_on_triple_fault ips = 10000000, = 1
megs: 16
vgaromimage: file = / usr / share / vgabios / vgabios.bin
vga: extension = vbe
floppya: 1_44 = 144cpm86.img, status = inserted
ata0: enabled = 1, ioaddr1 = 0x1f0, 0x3f0 = ioaddr2, irq = 14
ata0-master: type = disk, path = "c.img", mode = flat, cylinders = 16 heads = 16, spt = 63
boot: floppy
floppy_bootsig_check: disabled = 1
log: / dev / null
panic: action = ask
error: action = report
debugger_log: -
vga_update_interval: 300000
keyboard_serial_delay: 250
keyboard_paste_delay: 100000
mouse: enabled = 0
private_colormap: enabled = 0
USER_SHORTCUTS: keys = f10
i440fxsupport: enabled = 0
```

Bochs is started in the created directory. The call to *bochs* without options shows version and a menu to. The *-q* option suppresses this configuration menu and start the emulation directly.

```
Host ~ $ bochs-q
```

Other

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware. Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems.

<<< | # # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_Bochs"

This page has been accessed 4147 times. This page was last updated on 27 September 2010 at 06:36 clock changed. Content is available under GNU Free Documentation License 1.2 .

FAUmachine 20090512 download installation emulate virtualization

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / FAUmachine]])

<<< | # # # | >>> | English

Contents

- 1 FAUmachine 20,090,512
 - 1.1 Installation
 - 2.1 Quick Start
 - 1.3 Images
 - 1.4 Subsequent

FAUmachine 20090512

Website: <http://www3.informatik.uni-erlangen.de/Research/FAUmachine/>

FAUmachine is both a hardware emulator and a virtualization solution. FAUmachine is open source and runs as a normal user process under Linux (x86). Since no kernel modules are used to accelerate the execution speed is low. Ports on OpenBSD and Windows are currently being developed. FAUmachine uses parts of the source code of QEMU. The advantages of FAUmachine are:

- It is possible to emulate, similar to QEMU, hardware error.
- For automated experiments, tests and installations is an experiment controller available.

Installation

On Ubuntu, the installation is done with a command line.

```
Host ~ $ sudo apt-get install FAUmachine
```

FAUmachine not as a package is available, so the sources are downloaded from the website and according to the installation instructions in the files *INSTALL* (Linux), *README.bsd* (BSD) or *README.macosx* (Mac OS X) to compile.

Quick Start

The script is started with the FAUmachine *faum*. This can only manage a virtual machine under the FAUmachine.

```
Host ~ $ faum
```

When you first start a wizard helps to create the virtual machine. This directory is selected, are stored in the files for the virtual machine. Thereafter, the image size is set in MBytes. This image is generated in the port. In the last step, the size of the memory is defined (64, 128 or 256 MB). These requirements are the *~/.Faumrc* stored in. Would like to change the stored values, so this file is to edit. The script calls the program itself *faum faum end-node* on the pc. After a short time, the main window. To install the operating system on the virtual machine, you simply select the image of the installation CD. To do this click on the *Insert* button under *hdc* (right top). *Power* can now use the button on the virtual machine booted and the guest system to be installed. The mouse and keyboard [Alt] + with the key combination [Ctrl] + [Esc] released. A special feature of FAUmachine is simulating hardware errors. These enable the menu item *Inject the Show List*. In the window you select the desired hardware failure. This can be tested as hardware testing programs.

Images

FAUmachine can only read images in raw format.

```
Host ~ $ source ~/.Faumrc
Host ~ $ qemu-img info $ IMAGE_PATH/node.def/ide_gen_disk-11.media
image: / VMS/faumachine/node.def/ide_gen_disk-11.media
file format: raw
virtual size: 2.0G (2147483648 bytes)
disk size: 2.0G
```

This image is in raw format can be of QEMU / KVM used directly.

```
Host ~ $ source ~/.Faumrc
Host ~ $ qemu-snapshot $ IMAGE_PATH/node.def/ide_gen_disk-11.media
```

It is also possible with *qemu-img* for FAUmachine to create images or convert it.

```
Host ~ $ source ~/.Faumrc host ~ $ qemu-img convert-O raw Platte.img \ $ IMAGE_PATH/node.def/ide_gen_disk-11.media
```

To save space, the use of sparse files is possible.

```
Host ~ $ source ~/.Faumrc
Host ~ $ dd if = / dev / zero \
of = $ IMAGE_PATH/node.def/ide_gen_disk-11.media \
bs = 1024k count = 1 seek = $ MEDIA_SIZE
```

These raw file of 2 GB only occupies one MB. an operating system is installed on that image, the file size grows.



FAUmachine with Ubuntu Server as a guest.



FAUmachine - Dialog windows error injection.

```
Host ~ $ source ~/.Faumrc
Host ~ $ qemu-img info $IMAGE_PATH/node.def/ide_gen_disk-11.media
image: /Vms/faumachine/node.def/ide_gen_disk-11.media
file format: raw
virtual size: 2.0G (2148532224 bytes)
disk size: 1.0M
```

Other

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware. Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems.

<<< | # # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulator/_FAUmachine"

This page has been accessed 2565 times. This page was last updated on 27 September 2010 at 06:36 clock changed. Content is available under GNU Free Documentation License 1.2 .

Gxemul 0.6.0 ARM MIPS PowerPC SuperH hardware emulator download installation

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / gxemul]])

<<< | # # # | >>> | English

gxemul

Website: <http://gxemul.sourceforge.net/>

Gxemul is a hardware emulator, to emulate the following computer architectures: *ARM*, *MIPS*, *PowerPC*, and *SuperH*. Gxemul is a complement or alternative to *qemu-system-arm*, *qemu-system-mips*, *mipsel qemu-system-and qemu-system-ppc*. Gxemul is open source and can be installed on most Unix-like operating systems. To install on Debian / Ubuntu package is the use of *gxemul*.

```
Host ~ $ sudo apt-get install gxemul
```

If there is no current package, the installation with the following steps are possible:

```
Host ~ $ wget \
http://gxemul.sourceforge.net/src/gxemul-0.6.0.tar.gz
Host ~ $ tar zxvf gxemul-0.6.0.tar.gz
Host $ cd ~ gxemul-0.6.0
Host ~ $ ./Configure
Host ~ $ make
Host ~ $ sudo cp gxemul /usr / local / bin /
```

For FreeBSD and its derivatives, such as PC-BSD, the installation is performed using the following commands.

```
Host ~ # cd / usr / ports / emulators / gxemul
Host ~ # make install clean
```

Started with the command *gxemul gxemul*. The *option-h* shows the many options of gxemul.

```
Host ~ $ gxemul-h
```

The *-H* option will list all the emulated machines.

```
Host ~ $ gxemul-H
```

Gxemul used images in raw format. Other image formats must be converted *img gxemul* with *qemu-by*. QEMU can embed images of gxemul While booting will usually fail because of the differences in the emulated hardware. Gxemul QEMU MIPS has a special mode, can be used to run virtual machines emulated by QEMU architecture. Using the URL <http://wiki.qemu.org/Download> is a minimal Linux, the MIPS architecture for use under QEMU is available that can be used gxemul of it.

```
Host ~ $ wget http://wiki.qemu.org/download/mips-test-0.2.tar.gz
Host ~ $ tar zxvf mips-test-0.2.tar.gz
```

The following command starts gxemul with this system.

```
Host ~ $ gxemul qemu_mips-x-M 128-o \
'Console = ttyS0 rd_start = 0x80800000 = 10000000 rd_size init = / bin / sh' \
0x80800000: mips-test/initrd.gz mips-test/vmlinuz-2.6.18-3-qemu
```

In the following example, NetBSD (DECstation 5000/200) installed under gxemul. First, *dd* the virtual disk created with. Then you download the image of the installation CD.

```
~ $ Dd if = / dev / zero of = bs = 1024 nbsd_pmax.img \ host count = 1 seek = 3000000 host ~ $ wget ftp://ftp.netbsd.org/pub/NetBSD/iso/4.0/pmaxcd-4.0
```

The configuration of the virtual machines via gxemul launch options. After starting the virtual machine will start the installation of NetBSD.

```
Host ~ $ gxemul-X-e-d 3max nbsd_pmax.img-db: pmaxcd-4.0.iso
```

After the installation options are applied.

```
Host ~ $ gxemul-X-e-d 3max nbsd_pmax.img
```

The installation of additional guest systems is at the URL <http://gxemul.sourceforge.net/gxemul-stable/doc/guestoses.html> described.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_GXemul "

This page has been accessed 3592 times. This page was last updated on 13 May 2010 at 16:42 clock changed. Content is available under GNU Free Documentation License 1.2 .



Mips Linux on gxemul.



The gxemul console.



NetBSD (DECstation 5000/200) under gxemul.

Linux Containers (LXC), OpenVZ templates

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / LXC]])

<<< | # # # | >>> | English

Contents

1 Linux Containers (LXC)
1.1 Installation
2.1 Creating containers
1.2.1 An isolated application
1.2.2 The configuration file LXC
1.2.3 Linux in the container
1.2.3.1 lxc-busybox
1.2.3.2 Problems
Ubuntu LTS 1.2.3.3 10.04.1 (Lucid Lynx) in container
1.2.3.4 Ubuntu 9.10 in the container
1.2.3.5 Fedora in the container with lxc-fedora
1.2.4 The user-space emulation QEMU
1.3 Additional commands
1.4 connection to a container
1.4.1 ssh
Vnc 1.4.2
1.4.3 VT

Linux Containers (LXC)

Website: <http://lxc.sourceforge.net/>

Linux Containers (LXC) is an OS-level virtualization solution for Linux. The application is server consolidation. When virtualization at the OS level, divide the host and guest systems with kernel (single kernel image - SKI). Here, the virtualized operating system and not the hardware. The host operating system generates more instances of itself this reason only guest systems with the same kernel as the host system supported. Since the overhead is low, the execution speed is very high. LXC addition, there are still OpenVZ on Linux and Linux-VServer as solutions for virtualization at the OS level. However, only LXC was included in the Linux kernel version 2.6.27. That is, for LXC is not a kernel patch necessary. We recommend a kernel version 2.6.29.

Installation

Under Ubuntu 10.04 lxc is to install the package. Furthermore, the package *bridge-utils debootstrap lxc libcap2 bin* necessary.

```
Host ~ $ sudo apt-get install bridge-utils debootstrap lxc libcap2-bin
```

In Fedora as the host system is a container with *febootstrap* created with Fedora.

```
Host ~ $ su-c "yum install bridge-utils lxc febootstrap debootstrap"
```

The LXC version command shows *lxc-version* of the.

```
Host ~ $ lxc version
lxc version: 0.7.2
```

For the allocation of system resources to the containers are Control Groups (cgroups) was used. when you install the packages in the */etc/fstab* not line *none / dev / cgroup cgroup defaults 0 0* If registered, was to activate the Control Groups using the following commands:

```
Host ~ $ sudo mkdir / dev / cgroup
Host ~ $ sudo sh-c "echo 'none / dev / cgroup cgroup defaults 0 0'>> / etc / fstab"
Host ~ $ sudo mount-a
```

The command-LXC LXC *checkconfig* kernels, the support of the review.

```
~ $ Host-lxc checkconfig
Found kernel config file / boot/config-2.6.32-22-generic

--- --- Namespaces
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Network namespace: enabled
Multiple / dev / pts instances: enabled

Control groups --- ---
Cgroup: enabled
Namespace cgroup: enabled
Cgroup device: enabled
Cgroup sched: Enabled
Cgroup account cpu: enabled
Cgroup memory controller: enabled
Cgroup cpuset: enabled

--- Misc ---
Veth pair device: enabled
Macvlan: enabled
Vlan: enabled
File capabilities: enabled
```

To configure the virtual network with *bridge-utils* is the network manager to remove it (see also http://qemu-buch.de/d/Netzwerkoptionen/_Virtuelle_Netzwerke_konfigurieren).

```
Host ~ $ sudo apt-get remove - purge network-manager network-manager-gnome
Host ~ $ service networking stop
```

The file */etc/network/interfaces* should be adapted. A backup copy must be created.

```
Host ~ $ sudo cp / etc / network / interfaces / etc / network / interfaces.bak
```

The entries that must be changed to *eth0* the past. The bridge and therefore the host computer in this example are accessible via the IP address 192.168.1.220.

```
# / Etc / network / interfaces
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet manual
```

```
auto br0
iface br0 inet static
    address 192.168.1.220
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.1
    bridge_ports eth0
    bridge_stp off
    bridge_maxwait 5
    post-up /usr/sbin/brctl br0 setfd 0
```

If the IP address assigned by DHCP, this configuration is used:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet manual
auto br0
iface br0 inet dhcp
bridge_ports eth0
bridge_stp off
bridge_maxwait 5
post-up /usr/sbin/brctl br0 setfd 0
up echo "nameserver 192.168.0.1">>> /etc/resolv.conf
```

Then, the network service is restarted.

```
Host ~ $ sudo /etc/init.d/networking restart
```

To test the configuration of the computer must be rebooted.

```
Host ~ $ sudo reboot
```

Creating containers

Containers allow a partitioning of the starters in their processes. This is similar to a chroot environment (see http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulatoren/_chroot). However LXC provides better insulation and configuration.

An isolated application

A container must not include an operating system. It is possible for an application, in this example, `/bin/bash`, run in a container. The purpose of the `execute` command `lxc`. After `the-n` container is the name of the state.

```
Host ~ $ sudo lxc-execute-n test01 /bin/bash
Host ~ # ps aux
USER PID% CPU% MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.1 1888 476 pts/0 S 17:37 0:00 /usr/lib/lxc/lxc-init - /bin/bash
root 2 0.5 1.0 5780 3084 pts/0 S 17:37 0:01 /bin/bash
root 29 0.0 0.3 2708 1020 pts/0 R+ 17:42 0:00 ps aux
Host ~ # exit
```

The LXC configuration file

With a configuration file can be configured including the network. The following is a supplied sample configuration file is used. The syntax of the configuration file explains the command `lxc.conf` you.

```
Host ~ $ sudo lxc-execute-n test1 \
-F /usr/share/doc/lxc/examples/lxc/macvlan.conf-bin /bin/bash
```

It is recommended to provide for each container its own directory.

```
Host ~ $ sudo mkdir-p/lxcontainers/tests
Host ~ $ cd ~/lxcontainers/tests
```

This directory is the configuration file `ip-static.conf` created with the following content. For the static IP address to an unused IP address in a private network is selected.

```
lxc.utsname = Test02
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.ipv4 = 192.168.1.150
```

The configuration file is specified with `the-f`.

```
Host ~ $ sudo lxc-execute-n Test02-f /lxcontainers/tests/ip-static.conf /bin/bash
Host ~ # ifconfig
    eth0 Link encap: Ethernet HWaddr 4a:49:43:49:79:bd
    inet addr: 192.168.1.150 Bcast: 0.0.0.0 Mask: 255.255.0.0
...
Host ~ # exit
Host ~ $
```

Linux in the container

Using a directory with a Linux file system and a configuration file is like the command `lxc-create` container with a Linux system to create a. A kernel is not needed, since among LXC share the guest systems the kernel with the host system. It is on the kernel version of the host system is not flexible. It is possible to operate them to the host system different Linux distributions on LXC.

Linux file systems can be created in different ways. For example, to copy the directories and files of an existing Linux system. It can also be used template from OpenVZ. Comfortable can be Linux file systems with the tools `lxc-busybox`, `debootstrap` (Debian / Ubuntu), `lxc-debian` (Debian) or `lxc-fedora` (Fedora) create. `Chroot` can be part of the Linux file systems to adapt. For example, the necessary software packages are installed. We recommend the installation of an SSH server for remote access.

An adaptation of the Linux file systems is also necessary for other reasons. It should be altered to some scripts to initialize or remove it. This applies, for example files in the `/etc/init` and `/etc/inittab`. Depending on the distribution for some mount points must be designed and necessary adjustments to some other configuration files.

In the container does not `udev`. `Udev` is used to manage devices connected to the operating current in the (`hotplug`). This information will be removed from the `sysfs` file system and device file in the `/dev` directory created one. `Udev` in the container system must be uninstalled. Instead, the appropriate device files in the `dev` directory created *manually*. The generation of devices can be included in any script. This `lxc` the file `/usr/local/bin/` with the following content `config`.

```
# / Bin / bash
#
#/ / Usr / local / bin / lxc-config
#
# Bodhi.zazen 's lxc-config
# Makes default devices needed in lxc containers
# Modified from http://lxc.teegea.net/
```

```

ROOT = $ (pwd)
DEV = $ {ROOT} / dev
if [$ ROOT = '/'] then
    printf "\ 033 [22; 35m \ NDO NOT RUN ON THE HOST NODE \ n \ n"
    tput sgr0
    exit 1
fi
if [ ! -D $ DEV] then
    printf "\ 033 [01; 33m \ nruns this script in rootfs \ n \ n"
    tput sgr0
    exit 1
fi
rm-rf $ {DEV}
mkdir $ {DEV}
mknod-m 666 $ {DEV} / null c 1 3
mknod-m 666 $ {DEV} / zero c 1 5
mknod-m 666 $ {DEV} / random c 1 8
mknod-m 666 $ {DEV} / urandom c 1 9
mkdir-m 755 $ {DEV} / pts
mkdir-m $ 1,777 {DEV} / shm
mknod-m 666 $ {DEV} / tty c 5 0
mknod-m 666 $ {DEV} / tty0 c 4 0
mknod-m 666 $ {DEV} / tty1 c 4 1
mknod-m 666 $ {DEV} / tty2 c 4 2
mknod-m 666 $ {DEV} / tty3 c 4 3
mknod-m 666 $ {DEV} / tty4 c 4 4
mknod-m 600 $ {DEV} / console c 5 1
mknod-m 666 $ {DEV} / full c 1 7
mknod-m 600 $ {DEV} / p initctl
mknod-m 666 $ {DEV} / ptmx c 5 2
exit 0

```

The script must be set executable.

```
Host ~ $ sudo chmod u + x / usr / local / bin / lxc-config
```

It will run on demand in the file system of the container (see below).

lxc-busybox

With the shell script *lxc-busybox* can create a small Linux system itself. Furthermore, a configuration file for LXC is generated. The options are the name (*-n*) and path (*P*).

```
Host ~ $ sudo lxc-busybox-n busy01-p / lxcontainers/busy01
empty password for root, do not forget to change it!
```

The configuration file is updated with the lines to the network configuration.

```

Host ~ $ sudo vi / lxcontainers/busy01/config

lxc.utsname = busy01
lxc.tty = 1
lxc.rootfs = / lxcontainers/busy01/rootfs
lxc.cgroup.devices.deny = a
# / Dev / null and zero
lxc.cgroup.devices.allow = c 1:3 rwm
lxc.cgroup.devices.allow = c 1:5 rwm
# Consoles
lxc.cgroup.devices.allow = c 5:1 rwm
lxc.cgroup.devices.allow = c 5:0 rwm
lxc.cgroup.devices.allow = c 4:0 rwm
lxc.cgroup.devices.allow = c 4:1 rwm
# / Dev / {, u} random
lxc.cgroup.devices.allow = c 1:9 rwm
lxc.cgroup.devices.allow = c 1:8 rwm
lxc.cgroup.devices.allow c = 136: * rwm
lxc.cgroup.devices.allow = c 5:2 rwm
# Rtc
lxc.cgroup.devices.allow c = 254:0 rwm
to complement # # Manual:
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.name = eth0
lxc.network.mtu = 1500
lxc.network.ipv4 = 192.168.1.0/24

```

The command *lxc-create* a container created.

```
Host ~ $ sudo lxc-create-f-n lxcontainers/busy01/config busy01
'Busy01' created
```

The command *ls* lists *lxc-container* at all.

```
Host ~ $ ls-lxc busy01
```

Informed about the status of a container of command *lxc-info*.

```
Host ~ $ sudo lxc-info-n busy01
'Busy01' is STOPPED
```

The system in the container, the command started *lxc-start*.

```

Host ~ $ sudo-start-n lxc busy01
init started: BusyBox v1.13.3 (Ubuntu 1:1.13.3-1ubuntu11)
udhcpc (v1.13.3) started
Sending discover ...
Sending select for 192.168.1.242 ...
Lease of 192.168.1.242 Obtained, lease time 21600
Please press Enter to activate this console.
BusyBox v1.13.3 (Ubuntu 1:1.13.3-1ubuntu11) built-in shell (ash)
Enter 'help' for a list of built-in commands.
/ Bin / sh: can not access tty; job control turned off
/ #

```

On a second console you check the status of the container.

```
Host ~ $ lxc-info -n busy01
'Busy01' is RUNNING
```

Access to the text console of the host system is using the command `console lxc`. More) is derr name of the container (*n*) and the console (-*t* be given.

```
Host ~ $ sudo lxc-console -n t 0 busy01
Type <Ctrl+a Q> to exit the console
busy01 login:
```

Shut down the containers with the command `lxc-stop`.

```
Host ~ $ sudo lxc-stop -n busy01
Host ~ $ lxc-info -n busy01
'Busy01' is STOPPED
```

If a container is no longer needed, he will command `lxc-destroy` destroyed.

```
Host ~ $ sudo lxc-n-destroy busy01
```

Problems

Sometimes appear on these error messages.

```
Host ~ $ sudo start-n lxc busy01
lxc-start: Device or resource busy - could not unmount rootfs old
lxc-start: failed to pivot_root to '/lxcontainers/busy01/rootfs'
lxc-start: failed to set rootfs for 'busy01'
lxc-start: failed to setup the container
```

The cause is a bug that a separate partition for `/var` experiencing. A detailed command output shows the following.

```
Host ~ $ sudo lxc-start -n 1 busy01 DEBUG-o $ (tty)
```

Ubuntu LTS 10.04.1 (Lucid Lynx) in container

In this example, the creation of the file system with `debootstrap`. Before a sub-directory for the container is created.

```
Host ~ $ sudo mkdir -p / lxcontainers / lucid / rootfs host ~ $ cd / lxcontainers / lucid
```

The file to be created `/lxcontainers / lucid / fstab` the mount points are defined. This file contains the following contents:

```
# / Lxcontainers / lucid / fstab
none / lxcontainers / lucid / rootfs / dev / pts devpts defaults 0 0
none / lxcontainers / lucid / rootfs / var / run tmpfs defaults 0 0
none / lxcontainers / lucid / rootfs / dev / shm tmpfs defaults 0 0
```

With `debootstrap`, the host system (64-bit installed).

```
Host ~ $ sudo debootstrap - arch amd64 sid \
/ Lxcontainers / lucid / rootfs http://archive.ubuntu.com/ubuntu
```

In `/lxcontainers / lucid / rootfs / lib / init / fstab` lines are with `/proc`, `/dev` and `/dev / pts` to uncomment that.

```
# <file System> <mount-point> <type> <options> <dump> <pass>
/ Dev / root / rootfs defaults 0 1
# None / proc proc nodev, noexec, nosuid 0 0
none / proc / sys / fs / binfmt_misc binfmt_misc nodev, noexec, nosuid, optional 0 0
none / sys sysfs nodev, noexec, nosuid 0 0
none / sys / fs / fuse / connections fusectl optional 0 0
none / sys / kernel / debug debugfs optional 0 0
none / sys / kernel / security securityfs optional 0 0
none / spu spufs gid = spu, optional 0 0
# None / dev Devtmpfs, tmpfs mode = 0755 0 0
none / dev / pts devpts noexec, nosuid, gid = tty, mode = 0620 0 0
none / dev / shm tmpfs nosuid, nodev 0 0
none / tmp none defaults 0 0
none / var / run tmpfs mode = 0755, nosuid, show through 0 0
none / var / lock tmpfs nodev, noexec, nosuid, show through 0 0
none / lib / init / rw tmpfs mode = 0755, nosuid, optional 0 0
```

Thus, the host system a name, for example, `horst`, receives, this is in `/lxcontainers / lucid / rootfs / etc / hostname` entered. Continues to receive the file `/lxcontainers / lucid / rootfs / etc / hosts` includes:

```
Horst 127.0.0.1 localhost
```

To further configuration changes to `chroot` in the applied environment. In this environment, install the package `ssh` for SSH access to this guest.

```
Host ~ $ sudo chroot / lxcontainers / lucid / rootfs / bin / bash guest ~ # apt-get install ssh
```

You put the Konfiurationsdatei `/lxcontainers / lucid / conf` content. The network settings are adjusted to the own distinct needs.

```
# / Lxcontainers / lucid / conf
lxc.utsname = horst
lxc.tty = 4
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.hwaddr = 4a: 49:43:49:79: be
lxc.network.ipv4 = 192.168.1.50
lxc.network.name = eth0
lxc.network.mtu = 1500
lxc.rootfs = / lxcontainers / lucid / rootfs
lxc.pts = 1024
#
lxc.cgroup.devices.deny = a
# / Dev / null and zero
lxc.cgroup.devices.allow = c 1:3 rwm
lxc.cgroup.devices.allow = c 1:5 rwm
# Consoles
lxc.cgroup.devices.allow = c 5:1 rwm
lxc.cgroup.devices.allow = c 5:0 rwm
```

```
lxc.cgroup.devices.allow = c 4:0 rwm
lxc.cgroup.devices.allow = c 4:1 rwm
# / Dev / {, u} random
lxc.cgroup.devices.allow = c 1:9 rwm
lxc.cgroup.devices.allow = c 1:8 rwm
lxc.cgroup.devices.allow c = 136: * rwm
lxc.cgroup.devices.allow = c 5:2 rwm
# Rtc
lxc.cgroup.devices.allow c = 254:0 rwm
```

Now, the container is created and started.

```
Host ~ $ sudo lxc-create-horst n / lxcontainers / lucid / conf
'Horst' created
Host ~ $ sudo lxc-start-n-d horst
```

It checks whether the virtual machine is running.

```
Host ~ $ sudo lxc-info-n horst
'Horst' is RUNNING
```

Access to the text console of the host system is using the command console lxc. Derr given name of the container (n) and the console (-t) is indicated.

```
Host ~ $ sudo lxc horst-console-n-t 0
Type <Ctrl+a Q> to exit the console
horst login:
```

Alternatively, log into an ssh.

```
Host ~ $ ssh root@192.168.1.50
```

Shut down the containers with the command lxc-stop.

```
Host ~ $ sudo lxc-stop-n horst
Host ~ $ lxc-info-n horst
'Horst' is STOPPED
```

Ubuntu 9.10 in the container

In this example, the creation of the file system with *debootstrap*. Before a sub-directory for the container is created.

```
Host ~ $ sudo-i
Host ~ # cd / lxcontainers
Host ~ # mkdir ubuntu.karmic-01
Host ~ # cd ubuntu.karmic-01
Host ~ # debootstrap - variant = minbase karmic rootfs
```

The file / etc / resolv.conf on the host system is copied to the file system.

```
Host ~ # cp / etc / resolv.conf rootfs / etc
```

The generation of devices is with the script above, /usr / local / bin / lxc-config done. In the directory of the file system of the container to be created it is running.

```
Host ~ # cd rootfs
Host ~ # / usr / local / bin / lxc-config
Host ~ # cd -
```

For the container name is a configuration file with the config.

```
# / Lxcontainers/ubuntu.karmic-01/config
lxc.utsname = hardy-01
lxc.tty = 4
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = br0
lxc.network.name = eth0
lxc.network.mtu = 1500
lxc.network.ipv4 = 192.168.1.0/24
lxc.rootfs = / lxcontainers/ubuntu.karmic-01/rootfs
lxc.cgroup.devices.deny = a
# / Dev / null and zero
lxc.cgroup.devices.allow = c 1:3 rwm
lxc.cgroup.devices.allow = c 1:5 rwm
# Consoles
lxc.cgroup.devices.allow = c 5:1 rwm
lxc.cgroup.devices.allow = c 5:0 rwm
lxc.cgroup.devices.allow = c 4:0 rwm
lxc.cgroup.devices.allow = c 4:1 rwm
# / Dev / {, u} random
lxc.cgroup.devices.allow = c 1:9 rwm
lxc.cgroup.devices.allow = c 1:8 rwm
# / Dev / pts / * - pts namespaces are "coming soon"
lxc.cgroup.devices.allow c = 136: * rwm
lxc.cgroup.devices.allow = c 5:2 rwm
# Rtc
lxc.cgroup.devices.allow c = 254:0 rwm
```

With the command chroot file system is the root-adjusted.

```
Host ~ # chroot / lxcontainers/ubuntu.karmic-01/rootfs
Host ~ # mount-t devpts devpts / dev / pts
Host ~ # mount-t proc proc / proc
Host ~ # mount-t sysfs sysfs / sys
```

The first step is the package gpgv Privacy Guard) to install GNU (.

```
Host ~ # apt-get install - force-yes-y gpgv
```

There are still following packages necessary.

```
Host ~ # apt-get update
Host ~ # apt-get install-y adduser apt-utils iproute netbase \
openssh-blacklist openssh-blacklist-extra openssh-server \
console-setup sudo ping
```

Described must be removed as *udev*.

```
Host ~ # apt-get remove - purge udev
Host ~ # rm -rf / etc / udev / udev lib /
Host ~ # apt-get
```

It is clear the following scripts to initialize.

```
Host ~ # cd / etc / init
Host ~ # rm * mountall upstart *
```

The password is set.

```
Host ~ # passwd
```

Leave the chroot environment.

```
Host ~ # umount / dev / pts
Host ~ # umount / proc
Host ~ # umount / sys
Host ~ # exit
Host ~ #
```

The network configuration file is processed:

```
/ Lxcontainers/ubuntu.karmic-01/rootfs/etc/network/interfaces

auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
    address 192.168.1.61
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.10
```

Further adjustments are necessary.

```
Host ~ # rm rootfs / etc / init / tty {5.6}. Conf host ~ # mkdir-p rootfs / var / run / network host ~ # touch rootfs / var / run / network / ifstate !
```

It is the following script to create. It is used to initialize the system in the container.

```
/ Lxcontainers/ubuntu.karmic-01/rootfs/etc/init/lxc.conf

# LXC - Fix init sequence to have LXC working with upstart
# Description "fix LXC container - Karmic"
start on startup
task
pre-start script
mount-t proc proc / proc
mount-t devpts devpts / dev / pts
mount-t sysfs sys / sys
mount-t tmpfs varrun / var / run /
mount-t tmpfs varlock / var / lock
mkdir-p / var / run / network
touch / var / run / utmp
chmod 664 / var / run / utmp
root.utmp chown / var / run / utmp
if ["$ (find / etc / network / upstart-type f-name)"] then
    chmod+x / etc / network / * / upstart || true
fi
end script
script
Home networking
initctl emit file system - no-wait
initctl emit local/filesystems - no-wait
initctl emit virtual-file system - no-wait
init 2
end script
```

The container is created and started.

```
Host ~ # lxc-create-f-n lxccontainers/ubuntu.karmic-01/config karmic-01
Host ~ # lxc-start-n karmic-01
```

On a second console connects you with the console of the container.

```
Host 2.Konsole ~ $ sudo lxc n-console-karmic-01
```

Alternatively, log into via *ssh* any computer on your network from.

```
~ $ Ssh root@192.168.1.61
```

The command *lxc-stop* system is shut down in the container.

```
Host 2.Konsole ~ $ sudo lxc-stop-n karmic-01
```

If the system is installed and configured to secure it. With the word archive can provide a template.

```
Host ~ # cd / lxccontainers /
Host ~ # tar-01.tar.gz czf ubuntu.karmic ubuntu.karmic-01
```

Im_Aufbau

Fedora in the container with lxc-fedora

In Fedora is the script to generate *lxc-fedora* Fedora containers available. The following example *fedora01* a container with the name generates.

```
Host ~ $ su - host ~ # mkdir-p mkdir-p / lxccontainers/fedora01 host ~ # cd / lxccontainers/fedora01 host ~ # fedora-create lxc What is the name for the
```

The system in the container, the command started **lxc-start**.

```
Host ~ # lxc-start -n fedora01
```

The user-space emulation QEMU

The user space emulation QEMU (see [# userspace emulation](http://qemu-buch.de/d/Spezielle_QEMU-Optionen)) can be combined with LXC. When the user space emulation is not a complete PC emulates, but it will not support the use of other dynamic libraries. Thus, programs can be started, which were actually compiled libraries of other architectures. QEMU is first installed. Furthermore, we need examples of libraries and binaries for various architectures. These can range from the QEMU website <http://wiki.qemu.org/Download> in point *QEMU Linux user mode emulation tests* to be downloaded.

```
Host ~ $ sudo apt-get install kvm qemu-qemu-kvm-extras
Host ~ $ sudo mkdir-p / lxccontainers / qemu-user-space emulation
Host ~ $ sudo chown `whoami` / lxccontainers / qemu-user-space emulation /
Host $ cd ~ / lxccontainers / qemu-user-space emulation /
Host ~ $ wget http://wiki.qemu.org/download/linux-user-test-0.3.tar.gz
Host ~ $ tar xzvf linux-user-test-0.3.tar.gz
Host $ cd linux-user-test-0.3
```

It is the file *userspace.sh* following content to mess with.

```
# / Bin / bash
/ Usr / bin / qemu-arm-L gnemu / qemu-arm / arm / uname-a
```

This script is to make it executable.

```
Host ~ $ chmod +x userspace.sh
```

The script is started in a container.

```
Host ~ $ sudo qemu lxc-execute-n-userspace01. / Userspace.sh
Linux 2.6.32-22-generic Schlappi # 33 SMP Thu Apr 28 13:27:30 Ubuntu UTC 2010 unknown armv7l
```

As you can see, it is the combination of LXC and the userspace emulation QEMU possible to use libraries and binaries the other processor architectures LXC.

Other commands

```
Host ~ $ sudo netstat-n-lxc busy01
Host ~ $ sudo ps-n-lxc busy01

lxc-checkpoint
lxc-netstat
lxc-setcap
lxc-unfreeze
lxc-wait
lxc-cgroup
lxc-ls
lxc-ps
lxc-unshare
lxc-checkconfig
lxc-freeze
lxc-monitor
lxc-restart
```

Connecting to a Container

ssh

vnc

```
Host ~ # echo '/usr/bin/vnc4server: 0-geometry 1024x768-depth 24' >> / lxccontainers / testus / etc / rc.local
guest ~ # apt-get install fluxbox vnc4se:
VT
<<< | # # # | >>>
```

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulator/_LXC"

This page has been accessed 3304 times. This page was last updated on 23 January 2011 at 19:46 clock changed. Content is available under GNU Free Documentation License 1.2 .

personal alpha for Windows (Alpha emulator), OpenVMS 8.3, dcl, license, download iso installation ssh decnet

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / personal alpha]])

<<< | # # # | >>> | English

Contents

- 1 Human alpha for Windows
 - 1.1 Installation
 - 2.1 OpenVMS 8.3 as a guest system
 - 1.2.1 Image installed with OpenVMS
 - 1.2.2 Installing OpenVMS
 - 1.2.3 Configuring OpenVMS
 - OpenVMS 1.2.4 Important commands

Personnel alpha for Windows

Download: <http://www.stromasys.ch/products/charon-axp/download-personalalpha/>

The emulator allows you to run personal alpha alpha software (OpenVMS, Tru64) on a standard PC with Microsoft Windows XP. personnel-alpha provides the host system 128 MB RAM, an Ethernet adapter, a VT terminal emulation and four virtual hard disks. For personal use personal alpha is free. For the equipment to be installed OpenVMS or Tru64 such licenses are necessary. The minimum system requirements are:

- Intel or AMD processor (1 GHz)
- 1 GB of RAM
- Ethernet Adapter
- 4 GB of free hard disk capacity.

Installation

To install click the downloaded file to the *PAlpha.msi*. If available on your PC no. NET Framework, it is installed in the first step. The installation of personal alpha is as for most installation programs on Microsoft Windows. That is, you must repeatedly press the Next button and then restart the PC.



Figure: personal alpha for Windows - Configuration

OpenVMS 8.3 as a guest system

VMS (Virtual Memory System) is an operating system the computer manufacturer Digital Equipment Corporation (DEC). It was at the time of publication (1978) an extremely advanced 32-bit operating system, multi-user and multitasking was capable. The company DEC was later acquired by Compaq, which in turn was acquired by Hewlett Packard. Meanwhile, VMS was renamed OpenVMS. run on OpenVMS on mobiles and SMS solutions in banking, stock market systems operate. Free OpenVMS licenses at the URL <http://www.openvms.hobbyist.com> available.

Image with installed OpenVMS

Download: ftp://es40nce:eqs438gfv@ftp.stromasys.com/openvms_83_nolic.tar.bz2

A virtual hard disk with installed OpenVMS / AXP 8.3 (without license) can be downloaded from the download URL and unpacked. The image file is *dka0.vdisk* under *C: Program Files \ Personal Alpha \ saved *. personnel-alpha is the *Start Menu, All Programs, staff alpha* called on. In *discs* one defines the downloaded image *dka0.vdisk* first virtual hard drive (*DKA0*). I delete the entry after *DKA100*. Under *Network* has to *ESA mapping to host adapter* the network card of the Windows computer behind. Desktop *Mapping Mode* choose *Shared*. With *Save* the configuration to ensure that machine. The name gives you an *node01*. The XML configuration is at *C: Program Files \ Personal Alpha \ node01.emu* saved **. The virtual machine is started with *Start*. A window with a prompt. At boot prompt, the boot process will start from the hard disk.

>>> Boot DKA0

At the login prompt, log into user as *SYSTEM* without password. To shut down, use the command *SHUTDOWN*.

\$ SHUTDOWN

Installing OpenVMS

To install the OpenVMS installation CD into the CD-ROM drive must be lodged.

personnel-alpha is the *Start Menu, All Programs, staff alpha* called on. Under the *Configuration* tab are the paths to the disks, ROM and floppy drives, CD set. First, a virtual hard disk can be created. This is the *make Empty Disk* button to click. In *Disk-Type*, virtual hard disks of different sizes are selected. This entire virtual machine to a DVD that fits, we choose *2007MB, rz28* out. As file names are added to *dka0.vdisk*. The image file is under *C: Program Files \ Personal Alpha \ saved *. In *discs* one defines the term *Image dka0.vdisk* first virtual hard drive (*DKA0*). I delete the entry after *DKA100*. Under *Network* has to *ESA mapping to host adapter* the network card of the Windows computer behind. Desktop *Mapping Mode* choose *Shared*. With *Save* the configuration to ensure that machine. The name gives you an *node01*. The XML configuration is at *C: Program Files \ Personal Alpha \ node01.emu* saved **. The virtual machine is started with *Start*. A window with a prompt.

>>>

The command *show dev* not make the devices display.

```
>>> Show dev
...
... DKA200 ... (DVD-ROM drive)
...
```

To boot from the DVD-ROM drive, the command is necessary:

>>> Boot DKA200

The installation program starts. Assistance can be obtained respectively with the *[?]*.

```
1) Upgrade, install or reconfigure OpenVMS Alpha Version V8.3
Enter CHOICE or? for help: 1
```

If the system should be re-installed *INITIALIZE* thereof.

Do you want to *INITIALIZE* or to *PRESERVE?* I

The hard disk is to be selected for installation.

```
Enter device name for target disk: (choices? For)?
... DKA0 ...
Enter device name for target disk?) For choices DKA0 (
```

The hard drive must be given a name.



Figure: personal alpha for Windows - Installation of OpenVMS

Enter volume label for target system disk: [ALFASYS] <Return>

It is recommended to format the disk with ODS-5. OpenVMS Version7.2 support from ODS-5. Furthermore, hard links should be made possible.

Do you want to initialize with ODS-2 or ODS-5? (2 / 5 /?) 5
Do you want to enable hard links? Yes

The system administrator is under OpenVMS system and requires a password.

Password for SYSTEM account: *****

The test installation will be a cluster member. Galaxy is a virtualization solution on OpenVMS, and is not used here.

Will this system be a member of an OpenVMS Cluster? No
Will this system be an instance in a OpenVMS Galaxy? No

To uniquely identify the DECnet network SCSNODE name is required.

Enter SCSNODE: node01

DECnet is not needed here.

Do you want to use DECnet? No

The SCSSYSTEMID (1-65535) must be unique in the network.

Enter SCSSYSTEMID: [65 534] <Return>

The next step, the time zone is set.

Select the number above that best represent the desired time zone: 20
Select the number above that best represent the desired time zone: 6
Is this correct? (Yes / No) [Yes] <Return>

It is to be selected between summer and winter time.

Is this time zone currently on daylight saving time?
No (Winter) | Yes (Summer)
Enter the Time Differential Factor [1:00] <Return>
Is this correct? (Yes / No) [Yes] <Return>

For each OpenVMS license key whose data are entered.

Do you want to register any Product Authorization Keys? <Return> First REGISTER a Product Authorization Key Product ... Enter one of the above choices

Entering the license keys to End 99.

Enter one of the above choices [1]: 99

All software packages can be selected for installation.

Do you want to install DECwindows Motif? [Yes]: <Return>
Do you want to install DECnet-Plus? [Yes]: <Return>
Do you want to install HP TCP / IP Services: [Yes] <Return>
Do you always want detailed description? [No]: <Return>
Do you want the defaults for all options? [Yes]: <Return>
Do you want to review the options? [No]: <Return>

In the end, again, the menu appears on the screen. It shuts down the system.

9) Shutdown this system
Enter CHOICE or? for help: 9

It returns to the boot prompt.

>>>

You press the Stop button in the personal alpha window. After the shutdown, the virtual hard disk are protected *dka0.vdisk*. With the copy, it is possible to restore the initial state of the virtual machine. Then you boot from the hard disk. This window is the Start button in the personal alpha-operate. At boot prompt, the boot process will start from the hard disk.

>>> Boot DKA0

At the login prompt, log into user system on top. To shut down, use the command SHUTDOWN.

\$ SHUTDOWN

Configuring OpenVMS

For the configuration must be as user SYSTEM login. The DCL commands (Digital Command Language) can be entered in uppercase or lowercase. For the network of queue managers is necessary. The command SHOW QUEUE / MANAGER to check if the queue manager is available.

\$ SHOW QUEUE / MANAGER

If no queue manager exists, one must be created.

\$ START / QUEUE / MANAGER / NEW

The TCP / IP configuration with the command.

```
$ @ SYS $ MANAGER: TCPIP $ CONFIG
HP TCP / IP Services for OpenVMS Configuration Menu
Configuration options:
  1 - Core environment
  2 - Client components
  3 - Server components
  4 - Optional components
  5 - Shutdown HP TCP / IP Services for OpenVMS
  6 - Startup HP TCP / IP Services for OpenVMS
  7 - Run tests
  A - Configure options 1-4
  [B] - Exit configuration procedure
Enter configuration option: A
```

```
1 - Core environment
HP TCP / IP Services for OpenVMS Core Environment Configuration Menu
Configuration options:
 1 - Domain
 2 - Interfaces
 3 - Routing
 4 - BIND Resolver
 5 - Time Zone
 A - Configure options 1-5
 [E] - Exit menu
Enter configuration option: A [Enter]
```

The network configuration can be done via DHCP if the network of the host system, a DHCP server is available. Otherwise, a fixed IP address on the network segment of the host system is given. After configuring the network and network services (SSH, FTP, ...) the network is started manually.

```
$ @ SYS $ STARTUP: TCPIP $ STARTUP
```

The network is tested with *PING*.

```
$ UCX
TCPIP> PING qemu-buch.de
TCPIP> EXIT
```

The trip is to check with the command.

```
$ UCX SHOW HOST qemu-buch.de
85.214.74.183 QEMU BUCH.DE
```

So the network until the next reboot is available, is an editor with the file *sys \$ startup: systartup_vms.com* adapt. In this example, the editor will *edit / edt* used. In Befehlsprompt the editor can be reached by *c* in the edit mode.

```
$ EDIT / EDT SYS $ STARTUP: SYSTARTUP_VMS.COM
* C
```

They are seeking the following lines and the comment character "!" remove. The dollar sign at the beginning of every line must remain.

```
$ ...
$ ENABLE AUTOSTART / QUEUES
$ ...
$ @ SYS $ STARTUP: TCPIP $ STARTUP
$ ...
```

With [Ctrl] + [z] takes you back into command mode. With the *exit* command to save your changes and exit the editor. If you give command instead *quit*, will not save the changes.

```
* EXIT
$
```

An error in editing is corrected, because the file system from the OpenVMS versioned files. The different versions can be identified by the numbers following the semicolon in the filename.

```
$ DIR SYS $ STARTUP: SYSTARTUP_VMS.COM
Directory SYS $ COMMON: [sysmgr]
SYSTARTUP_VMS.COM, 2 SYSTARTUP_VMS.COM, 1
Total of files second
```

To test the network configuration, OpenVMS is rebooted.

```
$ REBOOT
```

To create a user the following steps:

```
$ @ SYS $ EXAMPLES: ADDUSER.COM
User Name: knut
Full name: Knut Bruno
Password: *****
UIC group number [200]: [Enter]
UIC member number: 201
Account name: knut
Privileges [TMBPBX, NETMBX]: [Enter]
Login directory: [kissing] [Enter]
Login device [sys $ SYSDEVICE]: [Enter]
```

It is manual, the home directory of the user to create.

```
$ CREATE / DIRECTORY [kissing] / = OWNER_UIC knut
```

The password is amended as follows.

```
$ SET DEFAULT SYS $ SYSTEM
$ RUN AUTHORIZE
$ UAF> MODIFY user-name/PASSWORD = new_password
$ EXIT
```

With *LOGOUT* logs you off.

```
$ LOGOUT
```

Important OpenVMS commands

OpenVMS systems use a scripting language, DCL (Digital Command Language). Unlike the Unix shells, the syntax of the DCL commands is the same layout. For example, almost all the commands the qualifier / *ALL*, each of the detailed spending information. The DCL commands can be entered in uppercase or lowercase. Furthermore, the commands are abbreviated. The password is reset with the following command:

```
$ SET PASSWORD
```

The time is displayed with *time show*.

```
$ SHOW TIME
```

The terminal properties are listed with the command:

\$ SHOW TERMINAL

The root directory is the *Master File Directory* (MFD) and is referred to as *000 000*. User directories (home directories) *User File Directories* (UFD) is called. User directories are located directly below the MFD and *root directory* called *Default* or *login*. The current path is called a *default directory*. Directories are recognized by the *ending*:. And *DIR* with square [] or angle brackets included *first Paths* in the sub-directories are separated by a period. View Directory Contents:

\$ DIR

Change to the directory *dumbo*:

\$ SET DEFAULT [DUMBO]

Displays the current path:

\$ SHOW DEFAULT

Create the directory *texte*:

\$ CREATE / DIRECTORY [. TEXTS]

The following command creates a file in the subdirectory:

```
$ CREATE [. TEXTS] Text Document.txt
...
[Ctrl] + [z]
```

To delete a directory, it must be empty.

\$ DELETE .] TEXTS * ; *

Display the permissions of a directory.

\$ DIRECTORY / PROTECTION texte.dir

To delete a directory you have the right to delete that list have. This is not the case by default. With this command you are the right to delete the subdirectory.

\$ SET PROTECTION = OWNER: D TEXTE.DIR

To delete the directory.

\$ DELETE TEXTE.DIR; *

Often, the *EDT* editor *EDIT* / used. After calling the editor you get to the command line mode.

\$ EDIT / EDT file

The following command will get you out *EDIT* / *EDT* save without:

*** QUIT**

The following command provides assistance to the editor *EDIT* / *EDT*:

*** HELP**

The following command changes the screen mode of the editor *EDIT* / *EDT*:

*** C**

The following command changes the command mode of the editor *EDIT* / *EDT*:

[Ctrl] + [z]

The following command will get you out *EDIT* / *EDT* with saving the latest changes:

*** EXIT**

The settings for his own account effected in the script file *LOGIN.COM*. Example:

```
$ HOME
$ Write sys $ output "Welcome, to OpenVMS"
$ Set terminal / insert
$ Set terminal / page = 34
```

From a Unix machine, log into SSH with X-forwarding on an OpenVMS machine.

~ \$ Ssh-X @ dumbo node01

OpenVMS leads to the terminal DECterm via X-forwarding to the Unix machine.

\$ CREATE / TERM

Files are transferred via SFTP from a Unix machine for OpenVMS machine.

```
~ $ Sftp knut @ node01
sftp> put foo.txt
Uploading foo.txt to / SYS $ SYSDEVICE / DUMBO / foo.txt
sftp> quit
```

A SSH Key allows logging in without a password on the OpenVMS machine. The SSH key pair without passphrase is generated on Linux.

```
~ $ Ssh-keygen-t dsa
Generating public / private dsa key pair.
Enter file # in which to save the key (/ root / .ssh / id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

The public key must be converted (for example *mypc.mydomain.com*).

~ \$ Ssh-keygen-e-f ~ / .ssh / id_dsa.pub > mypc_mydomain_com.pub

File named *AUTHORIZATION*. A (note the period at the end of the name) is the following content to mess with:

KEY MYPC_MYDOMAIN_COM.PUB

Then you copy both files to the OpenVMS machine in the subdirectory *SSH2*

```
~ $ Sftp knut @ node01
kнут @ node01's password: ****
sftp> mkdir ssh2
sftp> cd ssh2
sftp> put AUTHORIZATION.
sftp> put mypc_mydomain_com.pub
sftp> quit
```

There is a login without password possible.

```
~ $ Ssh-X @ knut node01
Welcome to OpenVMS (TM) Alpha Operating System, Version V8.3
$
```

To login from a OpenVMS machine to a Linux machine without a password, OpenVMS is an SSH Key to generate. If the directory [*.SSH2*] does not exist, it must be created. With *SSH_KEYGEN* be in the [*.SSH2*] the two files *KEYNAME.KEYNAME.PUB* and generated.

```
$ @ SYS $ MANAGER: TCPIP $ SET DEF SYS $ DEFINE_COMMAND $ LOGIN $ CREATE / DIRECTORY [SSH2.] [. KEYNAME $ SET DEFAULT SSH2] $ SSH_KEYGEN [SSH2.]
```

IDENTIFICATION continue the file size. The following content to mess with.

```
IdKey keyname
```

The public key is using *sftp* or *scp* the directory *~./*) Of the target user on the Linux box to *ssh* to copy (for example:

```
$ SCP KEYNAME.PUB I @ pluto: / home / .ssh I /
```

On the Linux computer file and convert them to attach the file *authorized_keys* in.

```
~ $ Cd ~ / .ssh
~ $ Ssh-keygen-i-f keyname.pub>> authorized_keys
```

On the OpenVMS machine you test the login without a password.

```
$ Ssh me @ pluto
```

```
<<< | # # # | >>>
```

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulator/_PersonalAlpha "

This page has been accessed 7198 times. This page was last updated on 27 September 2010 at 06:37 clock changed. Content is available under GNU Free Documentation License 1.2 .

Solaris Zones container, create zonecfg add remove, boot zoneadm clone

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / Solaris Zones]])

<<< | # # # | >>> | English

Solaris Zones

Website: <http://www.sun.com/bigadmin/content/zones/index.jsp>

The Solaris operating system comes with the Solaris Zones own virtualization solution with the operating system level. In each case, a virtual system environment with its own file system and its own processes, users, and at least one network interface is created. Simply put, there are multiple operating systems run in parallel on a kernel. For example, here is the area generated *testuslongus*.

```
Host ~ # zonecfg-z testuslongus
zonecfg: testuslongus> create
zonecfg: testuslongus> set zonepath = /mnt/testuslongus
zonecfg: testuslongus> set autoboot = true
```

It adds a network interface. The network interface is the existing *rtls0*.

```
zonecfg: testuslongus> add net
zonecfg: testuslongus: net> set address = 10.0.2.16
zonecfg: testuslongus: net> set physical = rtls0
zonecfg: testuslongus: net> end
```

If the zone set, you check the configuration, save and exit the configuration tool.

```
zonecfg: testuslongus> verify
zonecfg: testuslongus> commit
zonecfg:> exit testuslongus
```

It creates a directory for the zone and secures it with appropriate access rights.

```
Host ~ # mkdir /mnt/testuslongus
Host ~ # chmod go-rx /mnt/testuslongus /
```

Thereafter, the system software is installed in the zone.

```
Host ~ # zoneadm-z install testuslongus
```

The system in the zone will start.

```
Host ~ # zoneadm-z boot testuslongus
```

Now you have a second system and logs on. The system in the zone is configured as usual, and it can import other software packages.

```
~ # Host-1 zlogin testuslongus root
```

To remove a zone again, you go down it, uninstall it and delete its directory.

```
Host ~ # zoneadm-z testuslongus containing
Host ~ # zoneadm-z testuslongus uninstall
Host ~ # rm-rf /mnt/testuslongus
```

<<< | # # # | >>>

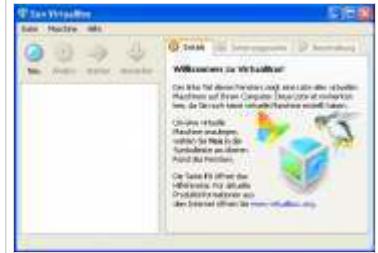
Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_Solaris_Zones "

This page has been accessed 3464 times. This page was last updated on 27 September 2010 at 06:37
clock changed. Content is available under GNU Free Documentation License 1.2 .

VirtualBox 4.0 Ubuntu Linux MacOS X Windows shared folder VBoxManage VBoxHeadless libvirt vboxwebsrv

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and Emulators / VirtualBox]])

<<< | # # # | >>> | English



VirtualBox - The Wizard to
create a virtual machine -
Step 1



VirtualBox - The Wizard to
create a virtual machine -
Step 2



VirtualBox - The Wizard to
create a virtual machine -
Step 3



VirtualBox - The Wizard to
create a virtual machine -
Step 4



VirtualBox - The Wizard to create a virtual machine - Step 5



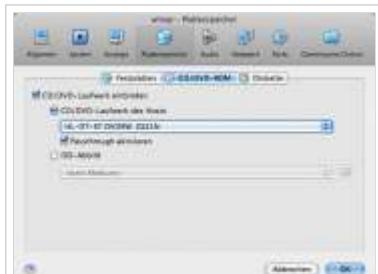
VirtualBox - The Wizard to create a virtual machine - Step 6



VirtualBox - The Wizard to create a virtual machine - Step 7



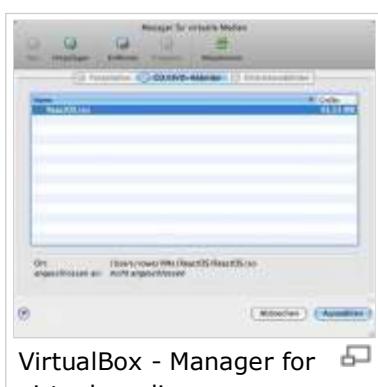
VirtualBox - The Wizard to
create a virtual machine -
Step 8



VirtualBox - The Wizard to
create a virtual machine -
Step 9



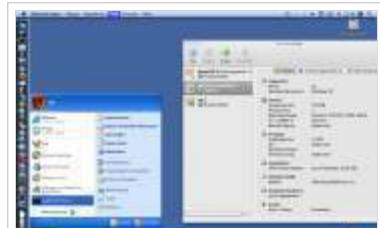
VirtualBox - Microsoft
Windows XP as a guest.



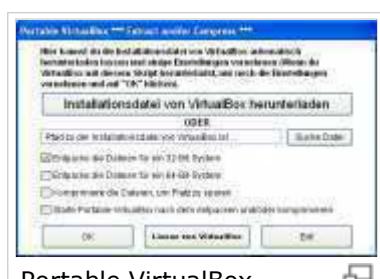
VirtualBox - Manager for
virtual media.

Contents

- 1 VirtualBox 4.0
 - 1.1 Installation
 - Linux 1.1.1
 - FreeBSD 1.1.2
 - 1.1.3 MacOS X
 - 1.1.4 Microsoft Windows
 - 1.1.5 OS / 2, eComStation
 - 2.1 Creating Virtual Machines
 - 1.3 Virtual Appliances
 - 4.1 Virtual Box Guest Additions
 - 1.4.1 Windows Guests
 - 1.4.2 Linux Guests
 - 1.5 control of the running virtual machine
 - 1.5.1 Seamless mode
 - 1.5.2 Assurance Points (snapshots)
 - 1.6 Configuration of the virtual machine
 - 1.6.1 General
 - 1.6.2 System
 - 1.6.3 Display
 - 1.6.4 Hard Drives, CD / DVD-ROM, floppy drive
 - 1.6.5 Audio
 - 1.6.6 Network
 - 1.6.7 Serial ports
 - 1.6.8 USB
 - 1.6.9 File sharing
 - 1.7 The menus
 - 1.7.1 File menu
 - 1.7.2 Machine menu
 - 1.7.3 Device menu
 - 1.7.4 Help menu
 - 1.8 VBoxManage, VBoxHeadless
 - 1.9 The libvirt library
 - 1:10 Other



VirtualBox - Seamless Mode (Windows XP on Mac OS X).



Portable VirtualBox.

VirtualBox 4.0

Download: <http://www.virtualbox.org/wiki/Downloads>

<http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.htm>

Documentation: <http://www.oracle.com/technetwork/server-storage/virtualbox/documentation/index.html>

VirtualBox by Oracle operates on the principle of Native Virtualization. Furthermore, full support Virtualization. The program is offered in two versions: the Standard Edition of VirtualBox with every feature under the proprietary *VirtualBox Personal Use and Evaluation License* (PUEL) and can be used free of charge for private purposes. VirtualBox Open Source Edition (OSE) has fewer features and is available under the GNU General Public License. Here it lacks the support for iSCSI, Gigabit NIC, USB and the Remote Desktop Protocol (RDP). VirtualBox can be installed under Linux, Mac OS X, Solaris (x86), Microsoft Windows, OS / 2, eComStation and FreeBSD. Furthermore, there is VirtualBox Portable (<http://www.vbox.me>) for Microsoft Windows XP, Vista and 7 This version does not come from Oracle and is therefore not officially supported. Portable VirtualBox works without installation and does not modify the registry. This makes it possible to copy with VirtualBox virtual machines on a USB stick and use it on another PC.

VirtualBox benefit of processors with hardware virtualization technology (Intel's Vanderpool, AMD's Pacifica). It can also be *nested paging / Rapid Virtualization Indexing* supported. For the guest systems SMP (Symmetric Multi Processing) allows up to 32 processors. Does the host system, no hardware virtualization technology, VirtualBox uses the "Raw Mode". This code is running as much as possible natively. Only the code that can not be implemented natively, will be run from an emulator which is based

on the code from QEMU. VirtualBox has a technique called "Patch Manager". Here, the code is analyzed and optimized at run time. Virtual disks can be emulated in so-called containers. These are called Virtual Disk Images (VDI). VirtualBox also supports disc image formats VMDK (VMware) and VHD (Microsoft). From version 2.2 supports the VirtualBox Open Virtualization Format (OVF). The OVF standard, the *Distributed Management Task Force* (DMTF) develops from. This makes it possible to use virtual appliances from across different virtualization solutions. The proprietary version can also embed objects as virtual iSCSI disks.

The Guest Additions enhance the integration between the host - and guest. For Microsoft Windows (Windows NT4 SP6a) and OS / 2 Warp, available as binary data (driver CD) before. For Linux as host system of the source code of the guest-addition in a virtual CD drive is mounted within the VM. The Guest Additions are under the proprietary license *PUEL*. For virtual machines with Microsoft Windows, Linux and Solaris OpenGL 2.0 support. In the experimental stage is to support Direct3D 8 or 9 for guest systems running Microsoft Windows. From version 3.1, as with QEMU / KVM, a live migration. Here is the feature *teleportation*. Furthermore, you can now go back to any snapshot. Microsoft-Windows-guest systems can use the 2D acceleration of the host system. The function of *seamless mode* applications run seamlessly in the host system directly to the desktop of the host system. As of version 3.2 can run under Mac OS X VirtualBox.

Installation

In the download URL are installation programs for use in different operating systems.

Linux

With most Linux distributions, VirtualBox Open Source Edition is included as a package. Under Ubuntu, the VirtualBox Open Source Edition with a command-line installs.

```
Host ~ $ sudo apt-get install virtualbox-ose
```

Better yet, the standard edition of the website to download and install. There are packages for common distributions. When you download the appropriate package, the package installer will open automatically. Where the installation is done via the command line, these commands are used:

```
Host ~ $ sudo apt-get install libqt4-opengl
Host ~ $ sudo dpkg-i virtualbox-*. deb
```

It is installing the group and create *vboxusers* module to compile kernel. Is invoked on the command *virtualbox* VirtualBox, *VirtualBox* or via the *Applications menu, system tools, VirtualBox*. By default both Linux virtual machines in the *~/.VirtualBox* stored as.

```
~ $ Host virtualbox
```

Or

```
Host ~ $ VirtualBox
```

Are there problems at the start, possibly not vote in the permissions. The user must be a member of the group *vboxusers*. This is made possible with the following command:

```
Host ~ $ sudo adduser `id-un` vboxusers
```

It needs to get in and then log back on. If after that is still different access rights helps to reboot. Test command is one's own group membership with the *groups*.

```
Host ~ $ groups ... vboxusers
```

After a kernel update, the VirtualBox kernel modules are reconfigured.

```
Host ~ $ sudo / etc / init.d / vboxdrv setup
```

FreeBSD

On FreeBSD the VirtualBox Open Source Edition is installed with the command line.

```
Host ~ # pkg_add-r virtualbox
Host $ cd /usr/ports/emulators/virtualbox
Host ~ $ make deinstall install config
```

MacOS X

Under MacOS X, the installation image Downloaded *VirtualBox *- OSX.dmg* after egebunden automatically. Simply click on the icon *VirtualBox* to the contents of the image display. To install, click on *VirtualBox.mpkg*. It starts a wizard. This is the license conditions and agree to select the hard drive in which to install. *VirtualBox* is started on the *VirtualBox* icon in the Application Folder (Programs). The host key (see below) is the left soft key [cmd].

Microsoft Windows

The Standard Edition of VirtualBox can be installed on Microsoft Windows XP, Vista and 7. To load file *VirtualBox-* Win.exe* down to and runs it. The settings in the *Custom Setup* are not altered. It should be noted that the installation be interrupted in existing network connections. After installing VirtualBox is the *Start Menu, All Programs*, called by *VirtualBox*.

This can be used VirtualBox Portable on a USB stick, the following steps are necessary. From the URL http://www.vbox.me/win_all.exe the file is *portable-virtualbox_v *- *- starter_v* and download to *C: Portable* to extract *\vbox*. *VirtualBox installation file download* button to click the program *C:\vbox\Portable\Portable-VirtualBox* is then *\VirtualBox.exe* to start and the *Portable*. After downloading the program should be discontinued. It is a USB flash drive with a capacity of 8 GByte stick to. The directory *C:\vbox-portable* USB flash drive is to copy the. The copied program is *VirtualBox.exe* Portable USB stick to run on the. It is the button *Find the file* path to the file *VirtualBox.exe* on the USB stick should be included. *Unzip the file system* is continued for a 32-bit system or *unzip the file for a 64-bit* enable. VirtualBox can now Portable from a USB key with portable *VirtualBox.exe* start. The so created virtual machines are stored on the USB stick. The virtual machines can be operated from a USB stick on other PCs with Microsoft Windows.

OS / 2, eComStation

For OS / 2 and eComStation an older version of the Open Source Edition has been ported. An installation guide can be found at the URL <http://www.akohl.net/os2/vbox/vbox.html> .

Creating virtual machines

For this example, Microsoft Windows XP chosen as the host system. The creation of virtual machines is done with a wizard. This is the *new* icon called on. First machine will be asked for the name of the virtual (*win xp*). Furthermore, the default operating system selected (*Windows XP*). Thereafter, the amount of RAM should be defined (512 MB). In the next dialog window, the size, type and name of the virtual disk is set. It is *produced* to enable *boot disk* and *hard drive*. As a data storage *medium* type is *dynamic growing* select. In *place* machine, the path for the new virtual set. The name of the created virtual machine appears in the list. To configure the virtual machine on, is to click on their name and select the appropriate options on the left side. By clicking on *mass memory*, or *CD / DVD-ROM* takes you to the dialog box to configure CD / DVDs. In *disk* installation disc, the machine is to make available to the virtual. This *drive* is *CD / DVD* to activate and *integrate CD / DVD drive*, select *the hosts*. With *pass-through* allows you to *activate* the virtual machine to access the CD / DVD drive on the host system. You start the virtual machine and install the operating system. The right [Ctrl] key and the mouse will leave the window of the virtual machine. The network configuration of the host system is via DHCP.

Virtual Appliances

Done) installed virtual machines (virtual appliances for VirtualBox from the URL <http://www.virtualboximages.com/vdi/index> download. For some documents are fees paid by some €. It can also use virtual hard disk from VMware and Virtual PC as a container under VirtualBox. It invites you on the URLs <http://www.vmware.com/appliances/> and <http://www.microsoft.com/downloads/> (search word: VHD) virtual machines VMware or Virtual PC for download and use the corresponding VMDK or VHD file as a container under VirtualBox. Including a new virtual machine must be created. Instead of generating a new virtual disk, the VMDK image is involved. The VMware tools are to remove the host system.

Virtual Box Guest Additions

It *addition*, the installation of *Virtual Box Guest* recommended. This is a CD image with different, optimized for virtual box, driver. This can be set as higher screen resolutions. Another advantage is the easy exchange of the mouse pointer between the desktop and the virtual machine window. Pressing the [Ctrl] key is no longer necessary. With the *Guest Additions*, the *seamless window mode* supported (see below). After an upgrade of VirtualBox Guest Additions have to be reinstalled.

Windows Guests

To install the *Virtual Box Guest Additions* is the virtual machine to start. You select the menu *item, equipment enhancements and install equipment*. The first call of this CD image is downloaded. If the CD image available to it the guest system is represented as the CD. By Auto-Run the setup program is called and the drivers are installed. Works do not start the car, is *VboxWindowsAdditions.exe* manually start the installation program. After installation, a reboot is necessary. The virtual CD is removed (*devices CD / DVD-ROM separate* menu). Then the screen resolution of the guest system is optimized.

Linux Guests

For many Linux distributions is the package *virtualbox-ose-guest-utils* available. It is easy to install in the Linux system in the virtual machine. Here is an example of the installation under Ubuntu as guest.

```
Guest ~ $ sudo apt-get install virtualbox-ose-guest-utils
```

Corresponding package is not available, the *Guest Additions* have to be compiled. Given these packages are necessary:

```
Guest ~ $ sudo apt-get install gcc make
```

It is the instant messaging device, select *Device install extensions*. Here, a virtual CD-ROM is inserted into the drive. It is to switch to the mount point. In Ubuntu, this is the directory */ media / cdrom*. Then, the installer is to start. Once installed, the virtual machine is restarted.

```
Guest ~ $ cd / media / cdrom
Guest ~ $ sudo ./VboxLinuxAdditions-x86.run
Guest ~ $ sudo reboot
```

Control of the running virtual machine

Seamless Mode

With the *Seamless mode* system, the desktop of the virtual machine to the desktop of the host integrated. This requires that the *Virtual Box Guest Additions* installed. It activates or deactivates the *Seamless Mode* from the menu *machine, Seamless mode* or using the key combination [host] + [L].

Backup Points (snapshots)

Securing points (snapshots) to save the states of the entire virtual machine (CPU, memory, devices and recordable discs). In this way, changes can be made to the system back. A backup point, the running virtual machine via the *machine menu, create backup point ...* created. If the virtual machine off points are the backup rider managed to *secure points*. In the dialog box the name of the backup point and a description should be given. Unfortunately, VirtualBox does not restore a backup point in the ongoing operation of the virtual machine, as this allows, for example, QEMU / KVM. Therefore, the host system to restore the state of a backup point of shutting down. The tab *securing points* is the desired anchor point with the right mouse button to select the shortcut menu and select *restore backup point*. Since backup points need much space, you no longer need protection points are deleted.

Virtual Machine Configuration

be off to configure the virtual machine must be it. Under the *Details* tab settings are the following possible.

General

It will be here the name of the virtual machine, system specifications and the path adapted to the securing points. Furthermore, the common clipboard of host and guest system is configured. It is possible to set copy direction of the clipboard: *Bi-directional*, *host to guest*, *guest to host* and *disabled*.

System

The main memory, the boot sequence, ACPI and IO APIC are adjustable under the *motherboard* tab. The tab is the number of emulated *processor* CPUs pretend (SMP). Furthermore, *NX PAE / activated*. The *Physical Address Extension*, 32-bit systems to address up to 64GB of RAM. Does the system host processor with hardware virtualization technology (Intel's Vanderpool, AMD's Pacifica), and let *VT-x/AMD-V Nested Paging* activated (tab *acceleration*).

Display

The tab *display* graphics memory is the size of the changeable. It can enable the 3D acceleration. VirtualBox supports remote control capabilities with the *VirtualBox Remote Desktop Protocol* (VRDP). The activation is done under the rider *remote control*. As a *guest* authentication method is to be specified when the access password should be one with. So that one machine from other computers on the network of the host system can reach the virtual, the *networks* as the interconnection *network bridge* under select. If necessary, the firewall should be adjusted. With a client for a remote desktop connection (RDPv5) connects you to the IP addresses of the host system.

Hard Drives, CD / DVD-ROM, floppy drive

From this point, let the guest system additional virtual hard drives, CD / DVDs and floppy disks to add. This is to click the icon with the plus sign. It can also activate additional controller (SATA, SCSI). New hard disks in the host system must be partitioned and formatted. On Microsoft Windows XP *diskmgmt.msc* is to call.

Audio

The sound output can be disabled or set a different sound card.

Network

It can be set up multiple network adapters. With *Adapter Network* adapter is a different select when the guest system with preset card has problems. There are these types of networks:

- *Not connected*: The virtual network cable is unplugged.
- *NAT (default)*: When Network Address Translation is internal IP address of the host system to the outside with the IP address of the host system to map the. The internal DHCP server assigns the guest system automatically assigns an IP address from 10.0.2.15. Also, the routing through the provided gateway (10.0.2.2) is defined. A network configuration for most operating systems is not necessary and access from the host system to the outside is now available. The firewall blocks all incoming instance of connections to this virtual network. Therefore it can not be accessed from outside the virtual machine. The host system is protected.
- *Network Bridge*: The virtual machine appears as an additional networked PC. VirtualBox creates an additional network card that provides a bridge to the network card of the host system. In the host system a static IP address is configured.
- *Internal network*: A communication is only possible with other virtual machines internally.
- *Host-only adapter*: The virtual machine directly accesses the NIC on the host system and receives the IP address from the DSL router or ISP. The host system is not protected.

Serial ports

Does the host system via serial interfaces, the host system can be made available. This is the point behind *Port mode host interface* selected.

USB

This system-on USB devices in the host system can access the host, *controller* is *USB-enabled active set*. For each of the guest system provided USB device filter is defined. To do this click on the icon with the plus sign. USB devices for which no filter was defined on, is in the running virtual machine via the menu *devices*, *USB devices*.

File sharing

With *common system folders* is an exchange of files between the host and guest support. This requires that the *Virtual Box Guest Additions* installed. A new *shared folder* is added to the icon with the green plus sign there too. For the safety of the folder may receive a write protection (*read only*). With the folder is *permanently produce* even after the restart. In the running virtual machine to enable a *common folder* on the *Device menu, common folder*.

On Microsoft Windows XP as host system selected in the Explorer menu item *Tools, Map Network Drive* one. To the right *folder*, click on *Browse*. In *VirtualBox Shared Folders* folder choose from. On Microsoft Windows 7, click on the Explorer icon in the taskbar. In the window, click on computer *network* and then having *VBOXSVR*.

Menus

File menu

- *Manager for virtual machines*: Here are virtual disks and ISO images managed. It can also be incorporated images of the VMDK VMware.
- *Appliance import*: a virtual machine in the Open Virtualization Format (OVF) is imported.
- *Appliance Export*: A virtual machine is turned off, the OVF format exported. This allows virtual machines to another host systems transfer. An exchange with VMware Player VMware Workstation 3 and 7 is possible.
- *Global Settings*: Here are directories for the virtual disks and configuration files changed. Furthermore, can the host key switch. If *Auto-snap mode keyboard enabled*, all keystrokes to the virtual machine be transferred if the window is active. VirtualBox is by default configured to automatically check for updates and displays a message in the presence of a newer version. It can change the language setting.

menu, press

This menu point will control the current virtual machine. The menu items will change if the virtual machine is started. Among other things can be here *full-screen mode [host] + [F]* and the *Seamless mode* (de-) activate. Furthermore, you can create backup points.

Devices menu

This menu appears only on the fly.

- *CD / DVD-ROM include*: Creates a virtual CD / DVD-ROM. This is either an image of it being accessed on the drive of the host system. Images have been added *for virtual media* to be a *manager* in advance.
- *CD / DVD-ROM disconnect*: Removes the virtual CD / DVD.
- *Floppy disk drive connector*: Creates a virtual disk. Images have been added *for virtual media* to be a *manager* in advance.
- *Floppy drive away*: Removes the virtual disk.
- *Network Adapter*: Draw the virtual network cable from the NIC or is it again.
- *USB devices*: Provides access to USB devices on the host system.
- *File sharing*: Configure folder, the host and guest system for data exchange to be used by the.
- *Remote control enable*: Enables the remote control with VRDP.
- *Install Guest Additions*: Specifies the CD image to the *Virtual Box Guest Additions* with one.

Menu Help

This menu points you get help. About *Content Manual* is the cost. Furthermore, VirtualBox allows you to register and test for updates.

VBoxManage, VBoxHeadless

VirtualBox contains the GUI and the command-line tool *VBoxManage* and *VBoxHeadless*. Most people use the VirtualBox GUI to manage virtual machines. Server but have no desktop environment. With *VBoxManage* manages one virtual machine on the command line. *VBoxHeadless* access allows the virtual machines over a remote connection. For example, lists the following command on all virtual machines.

```
Host ~ $ VBoxManage list vms
```

A virtual machine with the name, *win-xp* is started as follows. About RDPv5 one connects to this machine.

```
Host ~ $ VBoxHeadless startvm-win-xp "
```

Assistance is obtained using these commands.

```
Host ~ $ VBoxManage-help  
Host ~ $ VBoxHeadless
```

A guide can be found at the URL <http://www.howtoforge.de/blogroll/vboxheadless---virtuelle-maschinen-mit-virtualbox-3-1-x-auf-einem-debian-lenny-server/> . The command reference is found in *VirtuelBox User Manual*.

The libvirt library

Website: <http://libvirt.org>

libvirt, a virtualization layer between software and management tools. Thus, the development of management tools for all virtualization solutions that support this library possible. The library *libvirt* supported since version 0.7, QEMU, KVM, VirtualBox, VMware ESX, Xen, LXC Linux container system, OpenVZ Linux container system, and User Mode Linux. *Libvirt* is released as open source. For various Linux distributions would require new software packages are installed. The library can be compiled with Cygwin and Mingw under Microsoft Windows. More information can be found at the URL http://qemu-buch.de/d/Managementtools/_libvirt-Tools . Here is an example of a remote connection via SSH to a VirtualBox machine with the name *Pluto* and the *one I use*.

```
~ $ Virsh vbox-c + ssh: / / I @ pluto / session
```

Other

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware . Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems. Requires special features, such as live migration, emulation of other processors, API emulation, injections of hardware faults, debugging and manual settings of the system time is QEMU / KVM (additional) use.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_VirtualBox "

This page has been accessed 14,063 times. This page was last updated on 23 December 2010 at 06:17 clock changed. Content is available under GNU Free Documentation License 1.2 .

Microsoft Virtual PC 2007 SP 1, XP mode, Disk2vhd, eComStation 2.0

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / Microsoft Virtual PC]])

<<< | # # # | >>> | English



Microsoft Virtual PC 2007
New virtual machine
Step 1



Microsoft Virtual PC 2007
New virtual machine
Step 2



Microsoft Virtual PC 2007
New virtual machine
Step 3



Microsoft Virtual PC 2007
New virtual machine
Step 4



Microsoft Virtual PC 2007
New virtual machine
Step 5



Microsoft Virtual PC 2007
New virtual machine
Step 6



Microsoft Virtual PC 2007
New virtual machine
Step 7



Contents

- 1 Microsoft Virtual PC 2007 SP1
 - 1.1 Installation
 - 2.1 Creating Virtual Machines
 - ReactOS 1.2.1
 - 1.2.2 The XP-mode Microsoft Windows 7
 - 1.2.3 Microsoft Windows Trial
 - 1.2.4 eComstation demo live CD
 - 1.2.5 eComstation 2.0
 - 1.2.6 Physical-to-Virtual: Disk2vhd
 - 1.3 Virtual Machine Additions
 - 1.4 Control of the virtual machine
 - 1.4.1 Full Screen
 - 1.4.2 Pause, Reset, closing and shutting down
 - 1.4.3 Snapshots?
 - 1.5 Hardware configuration of the virtual machine
 - 1.6 Configuration of Microsoft Virtual PC 2007
 - 1.7 Control via Command Line
 - 1.8 Virtual disk management
 - 1.9 Other



Microsoft Virtual PC 2007 SP1

Download: <http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx>

Microsoft Virtual PC 2007 SP1 is a free virtualization software for Microsoft Windows XP, Vista, 7, Server 2003 and offered to Server 2008. Good integration is achieved on Microsoft Windows 7. As a guest systems officially Microsoft Windows versions, MS-DOS and OS / 2 are supported. It can still install many Linux distributions. Microsoft Virtual PC 2007 does not support snapshots, like QEMU / KVM. Originally, Microsoft Virtual PC x86 emulator for Mac OS X development. Since current Macs use Apple Intel processors, the development of Microsoft Virtual PC for Mac OS has been discontinued. Microsoft Virtual PC 2007 supports full virtualization, but does not depend on them. The Full Virtualization (see <http://qemu-buch.de/d/Grundlagen>) allows a much higher execution speed. The *Microsoft Hardware-Assisted Virtualization Detection Tool* can detect whether the hardware supports the Full Virtualization. Alternatively, with a Linux live CD to check this (see <http://qemu-buch.de/d/Installation>). It is necessary to ensure that support for hardware virtualization is enabled in the BIOS of the computer (example):

Advanced
CPU Configuration
Virtualization Technology [Enabled]



Installation

After downloading the installation program is started. In the beginning is the license agreement to accept and then enter the user name and organization. Microsoft Virtual PC 2007 available to all users or only the current user will be provided. The default installation path (*C:\ Program Files \ Microsoft Virtual PC*) need not be changed. When you install Microsoft Virtual PC 2007 wird short time, the network adapter, which is separate Internet connection. Associated warnings can be ignored. Microsoft will start Virtual PC 2007 über the *Start menu, All Programs, Virtual PC*.

To import the Microsoft Windows CD and DVD as an image, additional software is required. Most CD / DVD burning programs support importing from CD / DVDs. A free program is ImgBurn (<http://www.imgburn.com>). It loads the installer *SetupImgBurn_*.exe* from the site now and start it. After starting the program, choose *Create image file from disc* to ISO image to create a.

Creating virtual machines

The creation of virtual machines is done with a wizard. The wizard is the first launch of Microsoft Virtual PC 2007 runs automatically. You start the wizard on the *new* icon too.

ReactOS

Below are the steps to machine up a virtual instance on ReactOS (<http://www.reactos.org/de/> as a guest operating system described). ReactOS should be compatible with the kernel of Microsoft Windows NT. This makes it possible to use programs and drivers for NT and its successor, 2000, XP, Vista and 7. This is working on the reproduction of the programming interface (API), Win32, Win16, OS / 2, Java and DOS. The operating system is released under the GPL. Thus, it would be possible to get independent from Microsoft and free alternative to Microsoft Windows. The project is currently in alpha stage and so ReactOS is still not recommended for everyday use. First, it invites the image of the install CD downloaded from the website and unzipped it.

A virtual machine under Microsoft Virtual PC 2007 to create is to first select the option to *create virtual computer*. Then after the name of the virtual machine is required. In this example, the virtual machine called *ReactOS*. As the *Windows* operating system specification is selected *2000th* Subsequently, the amount of memory is to be specified. The proposal of the Wizards is not to take over. It is the point *adjustment of the working memory* activate and amount of RAM with 128 MB of set the. The amount of memory must not exceed half of memory in the host system. In the next dialog windows for the virtual disk is set. The virtual disk (Image) is a container file with the extension. *Vhd*. There is the option to select a *new virtual hard disk*. If the option selected to *enable disk-Undo* all changes to the virtual disk to a separate file (. *Vud*) written. The *VUD* files (Virtual Undo) are the overlay files of QEMU (see comparable http://qemu-buch.de/d/Speichermedien/_Images_anlegen). The wizard suggests half of free hard disk capacity of the host system as the value for the virtual hard disk. This suggestion of the wizard is not to take over. The size of the image should be a GB. Furthermore, the path is to pretend to this file. At the end, an overview is given of all options. If everything is OK, click on the button *Finish*. If all steps executed the wizard, the virtual machine name appears in the list of the main window of Microsoft Virtual PC 2007. To further configure the virtual machine, choose their name from, then click on the *Settings* button. Is chosen under *Network Adapters* for the *Shared Networking (NAT)* from. To start the virtual machine, choose its name from and then click on the *Start* button.

In the window of the virtual machine menu, *CD, ISO image, capture* the ISO image of the downloaded installation CD to choose. Installing ReactOS is started by clicking in the window of the virtual machine and press any key. First, the German keyboard is selected. Thereafter, the virtual disk is partitioned and formatted. The target directory for the installation of the ReactOS files need not be changed. After the boot loader is installed, the default option *Install bootloader on the harddisk (MBR) holds*. Then the virtual machine must be restarted. The installation continues graphically. To release the mouse pointer from the virtual machine window, press the [AltGr] firmly.

The XP-mode Microsoft Windows 7

Download: <http://www.microsoft.com/windows/virtual-pc/download.aspx>

For Microsoft Windows 7 (Professional, Enterprise and Ultimate) is the XP mode. The XP mode is a free image with a custom Microsoft Windows XP Professional. This is aimed at Microsoft Windows 7, a Windows (XP) compatibility. A license for Windows XP is included. The XP mode can not only run on Microsoft Virtual PC 2007th VMware Player and VMware Workstation can import the XP mode (see http://qemu-buch.de/d/Ahnung/_Weitere_Virtualisierer_und_Emulator/_VMware-Player). After you install Microsoft Virtual PC 2007 URL is the XP mode of the <http://www.microsoft.com/windows/virtual-pc/download.aspx> download. Then the XP mode by calling the file to install *WindowsXPMode_de-en.exe*. Where C is the proposed location: *\ Program Files \ Windows XP fashion over *. After agreeing to the license terms, the password for the XP guest system is to be specified. It is recommended that you enable *automatic updates*. On Microsoft Windows 7, the XP mode is integrated into the operation of the host system. This is the *Start menu, All Programs, Windows Virtual PC* on *Windows point-mode applications added XP*. In that folder will appear under all the XP mode, installed programs. This can be started as Windows 7 applications.

Microsoft Windows Trial

Microsoft provides the URL <http://technet.microsoft.com/en-us/bb738372.aspx> finished installed virtual machines with Microsoft Windows versions of Microsoft Virtual PC 2007 download for the. These machines are 30 days expiration. Other virtual machines can be found when one <http://www.microsoft.com/downloads/> enters a search term *VHD*. It downloads the archive of the desired virtual machine and unpack it. In Microsoft Virtual PC 2007 you click the *New* icon and selects the point in the wizard to *add existing virtual machine*. Subsequently, the path to the configuration file (. *Vmc*) to the downloaded virtual machine. The window *settings*, the virtual machine to be able to adapt to. After pressing *OK*, the virtual machine appears in the list.

eComstation demo live CD

Download: <http://www.ecomstation.com/democd/>

Microsoft Virtual PC 2007 supports exotic operating systems such as OS / 2 and eComStation. In this example, the eComstation demo live CD for the guest system is used. eComStation is a development of OS / 2 OS / 2 is a multitasking-capable, stable operating system for PCs. It was originally jointly developed as a successor to MS-DOS from IBM

and Microsoft. After Microsoft withdrew in 1991, IBM developed it further. IBM has been the development of OS / 2 abandoned. The successor eComstation is mainly developed by the Dutch company Mensys and through the open-source community Netlabs.org. OS / 2 and eComStation are still occasionally in banks, insurance companies and airlines to be found. eComStation is a lean operating system that runs on older hardware. In addition to the applications that were programmed for OS / 2 and eComStation, DOS can be operated and 16-bit Windows applications. With ODIN (<http://de.os2.org/projekte/odin/>) allows Microsoft Windows 95/98 and NT programs to run on eComStation directly. The graphical user interface eComstation consists of the *Workplace Shell* (WPS) and the *Presentation Manager*. It is object oriented. That is all the system components (drives, files, directories, printers, links to programs) are represented as an object and can be controlled with context menus and drag-and-drop. The virtualization of OS / 2 and eComStation is problematic because these systems use rarely used commands of the processor. The demo live CD of eComstation can be downloaded for free.

It is the first option select *create virtual computer*. In this example, the virtual machine *live CD* called *eComstation*. The operating system is *OS Benchmark 2 / selected*. It is the point *adjustment of the working memory* activate and amount of RAM with 256 MB of set the. In a live CD not a virtual disk is needed, Microsoft Virtual PC 2007 provides the ability but not on a virtual hard drive to give. Therefore, *hard drive*, the option to select a *new virtual*. The size of the image in this case does not matter. If all steps executed the wizard, the virtual machine name appears in the list of the main window of Microsoft Virtual PC 2007. To further configure the virtual machine, choose their name from, then click on the *Settings* button. Is chosen under *Network Adapters* for the *Shared Networking (NAT)* from. To start the virtual machine, choose its name from and then click on the *Start* button.

In the window of the virtual machine menu, *CD, ISO image, capture* the ISO image of the downloaded demo live CD of eComstation select. At the start of the demo live CD that default values should be selected. It seems the GUI of eComstation. To configure the network, you open the *web* folder on the desktop. Then click on the icon *Configure Internet Connection*. In the wizard choose the Ethernet controller and the next step is to configure the IP address. Then is to click *Activate Settings*. With the installed Firefox web browser to surf the Internet safely, because there is no malware for eComStation.

To release the mouse pointer from the virtual machine window, press the [AltGr] firmly, the virtual machine window is closed, a query is whether the state of the virtual machine to be stored. This makes it possible to store despite live CD settings. Also, the next launch of the virtual machine is faster.

eComstation 2.0

In this example eComstation 2.0 is installed as a guest. To install eComstation the paid installation media and registration keys are required.

It is the first option select *create virtual computer*. In this example, the virtual machine called *eComstation2*. The operating system is *OS Benchmark 2 / selected*. It is the point *adjustment of the working memory* activate and amount of RAM with 128 MB of set the. In the next dialog windows *hard disk* is the option to select a *new virtual*. The size of the image will be two gigabytes. If all steps executed the wizard, the virtual machine name appears in the list of the main window of Microsoft Virtual PC 2007. To start the virtual machine, choose its name from and then click on the *Start* button. In the virtual machine window is the menu-point *CD, Physics D: drive* to enable *use* for your installation CD is the start of.

In the boot menu of the installation CD that default values should be selected. Installation is done in graphical mode. First, the keyboard layout is selected. Thereafter, the license terms are acceptable. EComstation for newcomers is the *Easy-installation*, the best choice. Thereafter, the registration key is entered. The next step is to partition the hard drive. This window will open with a new *New Volume*. In the instant-messaging volume, *Create New* is created a new volume. This volume is *Create a bootable* activate. In the dialog box is behind the *letter*, the letter C to select and name must be specified as *Volume2*. Then is to click on *Next*. The window size as a *disk partitioning* is the value indicated 2047 and *partition* to enable *use primary*. The window is completed with *finish*. With the menu *system, Save Changes* to save these settings and then click on *Exit*. With *Next* you get to format the hard disk. As type choose *HPFS* and click on *format*. After formatting is to click on *Next*. It will select the country and time zone. The next step, the hardware components and the sound hardware can be determined. In step *networking networking Install support* is activated. After clicking *Next*, all network types (TCP / IP, Samba, NetBUI) enabled. *Workstation ID* then is behind the machine name and *domain LAN* behind "WORKGROUP" to indicate with *Next* and *Finish* is the first part of the installation completed and it will restart the virtual machine.

After the restart following the second part of the installation in the window *Final Tasks*. Below the tabs of this window configurations can be made. For example, the tab *screen*, the screen resolution set at. This only work after the restart. eComstation is on the menu *Ecs, shut shut down*. It should be saved in the host system Verzeichnismit the image (. Vhd) and the configuration file (. Vmc) to restore the original state with problems of the virtual machine. The installation CD is to be removed from the drive of the host. In the settings of the virtual machine is under *Network*, select *Shared Networking (NAT)* set up. Then the virtual machine is restarted with the *Start* button. In the virtual machine window is the menu *CD*, the *CD release*. For the configuration of the network is in the host system of the instant-messaging system setup, *Network Adapters and Protocols* to activate and then select *Configure MPTS*. As an adapter from *Realtek RTL8139* choose. The network configuration must be done via DHCP. Then a restart of eComstation is necessary.

Physical-to-Virtual: Disk2vhd

Download: <http://technet.microsoft.com/en-us/sysinternals/ee656415.aspx>

The tool supports conversion *Disk2vhd* a running Windows system in a virtual machine for Microsoft Virtual PC. The hard drive must be at most 127 GB in size. The transfer of an operating system from one physical machine to a virtual machine is called physical to virtual (see http://qemu-buch.de/d/Speichermedien/_Physical-to-Virtual). For a running operating system runs with applications and data without reinstalling old hardware into a virtual machine.

The system should be optimized to convert it. This includes the deletion of unnecessary garbage and cleaning the Windows registry. Furthermore, all startup entries are to be deactivated. The elimination of garbage can with the free program *CCleaner* (http://www.filehippo.com/download_ccleaner done). After installing and launching this program, click on *Analyze*. Then enable *Run Cleaner* in order to dispose of the junk file found. To clean the Registry, the Registry button and select *Check for errors* to click. After the analysis is activated to fix mistakes!. Thus, the virtual Windows startup no error messages produced are cut off prior to the move, the startup entries. After the move they have to activate again. Useful (this is the free program *Autorun* <http://technet.microsoft.com/de/de/syinternals/bb963902.aspx>). After installing and starting the program, choose the tab *autoruns.exe Everything*. To disable the entries are listed.

The tool *Disk2vhd* is installed, started and there are the partitions of the system selected. It is also the destination for the image file (. Vhd) indicated. The destination directory should have sufficient storage capacity to hold these files.

Virtual Machine Additions

If a Microsoft or OS / 2 as guest operating systems run Windows, *addition* is the installation of the *Virtual Machine* is recommended. This is a CD image with optimized drivers. This virtual machine is in the window of either the instant-messaging tools, *integration components* or *install* the menu item *Action, Virtual Machine Additions installed update / select*. It is included a CD image in the virtual CD-ROM drive. In the guest system CD, the *Virtual Machine Additions* from the virtual install. Microsoft Windows as guest system is usually the case when you insert the virtual CD automatically called in the file *setup.exe*. If it fails, the automatic start, so manually in the file manager the file *setup.exe* to start. After installation, the host system must be rebooted. To set a higher screen resolution in the guest system, just click the right mouse button on any free space on the Windows desktop. Under the *Settings* tab to configure the screen resolution. On Microsoft Windows 7 as the host system can be in the window of the virtual machine over the menu item *Tools, Settings*, configure the integration components. The *Virtual Machine Additions* support system to exchange data between host and guest on a common clipboard (*copy & paste*) and drag files (*drag & drop*).

Control of the virtual machine

Full screen

A switch to full screen mode using the key combination [AltGr] + [Enter] possible.

Reset, closing and shutting down

In the virtual machine window is the point of action the commands *stop, reset, closing* and *shutting down* available. The *Pause* command freezes the virtual machine. With instant-messaging campaign, resume running the machine on again. The *Reset* command similar to the reset button on a real PC. In this command, threatened loss of data. If the command is applied *Close* dialog box appears with the choices *off* and *save a state*. *Off* corresponds to pulling the power cable in a real PC. Here, too, threatened loss of data. *Save state* corresponds to the sleep mode on a laptop. It saves the state of the virtual machine before it is terminated. The next time the system of the last state is restored. If the *Virtual Machine Additions* installed, this command is available to *shut down*. For a clean shutdown of the host system is introduced. Was virtual disk *undo disks enabled* for the selected, the *Close* command are other options available. *Save For state and save changes*, the contents of the virtual file will undo. That is, the changes are not stored in the container file. In contrast, when *turning off and save changes*, the virtual machine shut down. *Turn off and delete changes* with the virtual file emptied and undo the changes are lost. The checkbox to change the virtual hard drive to take to file in the Virtual undo the saved file to change in the original container-written. It is recommended that you create when using Virtual undo files in each virtual hard disks for operating system and data.

Snapshots?

Microsoft Virtual PC 2007 supports any snapshots. I need a backup point, we stopped the virtual machine. Then you copy the directory of the virtual machine. If necessary, you open the copied virtual machine. To do this click on *New* and select the item in the wizard to *add existing virtual machine*. Subsequently, the path to the configuration file (. Vmc) to the copied virtual machine. The window *settings*, the virtual machine to be able to adapt to. After pressing *OK*, the virtual machine appears in the list.

Hardware configuration of the virtual machine

To configure the virtual machine must be shut down. It may also not be in a saved state. The corresponding dialog box calls you on the *Settings* button on.

- *File Name*: The name of the virtual machine can be changed.

- **Memory:** The amount of memory (RAM) of the virtual machine is configured here.
- **Hard Drive 1/2/3:** It can be configured VHD three to. This is the path to the container file (.Vhd) indicated. About *Wizard virtual disk* drive is a wizard to create a new virtual start.
- **Undo Disks:** If the option *Undo disks* are selected, changes to the virtual disk to a separate file (.) Vud written all.
- **CD / DVD drive:** CD / DVD drive is primary or secondary controller can be connected to the. Most of the secondary IDE controller is used.
- **Disk:** the host system detects the host system inserted disk if *disk* is activated *automatically detect*.
- **COM1 / 2:** Up to two serial interfaces to the host system will be made available to. These can be a serial port on the host system, a named pipe, or a text file to be assigned.
- **LPT1:** There may be a parallel port on the host system the host system will be made available to.
- **Network:** Up to four network cards to the host system will be made available to. In addition to the NIC of the host system an emulated network card of the type Intel is ready to put 21 140.
 - *NIC of the host system:* the host system receives full access to the network of the host system and is also accessible from the outside. The IP address will either be configured by an external DHCP server or set manually.
 - *Offline:* The virtual cable is a virtual network card deducted from.
 - *Local only:* via an internal virtual network, the virtual machines connected to each other. It is not linked to the host system is still possible to external networks.
 - *Shared Networking (NAT):* With Network Address Translation is the internal IP address of the host system to the outside with the IP address of the host system is shown. The internal DHCP server assigns to the host system an IP address from 192.168.0.0. Also, the routing through the provided gateway is defined. A network configuration for most operating systems is not necessary and access from the host system to the outside is now available. The firewall blocks all incoming instance of connections to this virtual network. Therefore it can not be accessed from outside the virtual machine. The host system is protected.
- **Sound:** The sound card can be de-) activate (.
- **Hardware Virtualization:** the appropriate hardware on the host system is at the hardware-assisted virtualization (Full Virtualization) to activate. Some host systems are incompatible with the hardware-assisted virtualization. Then, this option is deactivated.
- **Mouse:** If the mouse pointer in a virtual machine does not respond correctly, here is a different configuration to choose.
- **Shared Folders:** Have the *Virtual Machine Additions* installed in the guest system, allows the guest-system directories of the host system will be made available to. These appear as network shares in the host system. This *folder* is chosen to *share* the directory on the host system. This directory is a drive letter for the host system to specify. Every time share with this drive is included after each restart.
- **Display:** In addition to configuring the screen resolution here are the full-screen mode and display the menu and status bar can be activated.
- **Closing:** The closing of the virtual machine is connected with certain actions.

configuration of Microsoft Virtual PC 2007

The configuration of Microsoft Virtual PC 2007 is via the *File* menu, *Options*. It appears that the *Virtual PC option*.

- **When starting to recover:** If the *Start* option from *Virtual PC Restore* is enabled in the virtual computer, all at the end of Microsoft Virtual PC 2007 running virtual machine restarts automatically.
- **Output:** Here are the virtual machines assigned to certain portions of the processor power. the standard-setting to the virtual machine in the active window to assign more CPU time with%, the virtual machine in the foreground about 70 of the computing power made available. If you work often with multiple machines, here are other preferences.
- **Hardware Virtualization:** the appropriate hardware on the host system is at the hardware-assisted virtualization (Full Virtualization) to activate.
- **Full screen mode:** The default option is resolution on the host operating system of resolution on the guest operating system to adjust the selected. This full-screen setting to the host system is adapted.
- **Sound:** The sound of the virtual machines in inactive windows can be disabled.
- **News:** If you use the Microsoft Virtual PC 2007 and is familiar with no reference needed to enable *Messages None*.
- **Keyboard:** The host key [AltGr] can be changed, in which one desired key combination in the field *Current* enters the *host key*. When set to *Windows shortcut* with *Allow* you choose a place such as system hotkeys are to be applied.
- **Mouse:** If the mouse pointer in a virtual machine does not respond correctly, here is a different configuration to choose.
- **Security:** Here can be certain features for users without admin rights block.
- **Language:** Changing the language setting is only after you restart Microsoft Virtual PC 2007 active.

Control via Command Line

The virtual machines can also *Virtual PC.exe* on the command line of the host system controlled. The / help shows all options.

```
Host C: \> cd C:\Program Files\Microsoft Virtual PC Host C:\> Virtual PC.exe / help-help - Displays information about command line arguments to-f
```

The following example *ReactOS*, the virtual machine with the name started in full screen mode.

```
Host C: \> Virtual PC.exe-pc-fullscreen ReactOS
```

Virtual disk management

On Microsoft Windows 7 can manage virtual disks like real hard drives. For this, open Disk Management.

```
Host C: \> diskmgmt.msc
```

instant-messaging *campaign*, *virtual hard disk*, with one puts a new virtual disk. Adjust its path, file name and size is to be specified. Furthermore, the format is selected:

- **Dynamically extensible:** The virtual disk occupies on the host system only as much space as it has data. With the level increasing the file size. It is necessary to ensure that the filled image in the host system has enough space.
- **Fixed size:** The file system of the host supports this kind of start the whole place. Access is somewhat faster than the dynamic format.

After creating the virtual disk it must be partitioned and formatted. To do this click the right mouse button on the *Unallocated* area. In the context menu choose *Datenträgerinitialisierung*. Then choose *New Simple Volume*. It takes the proposed partitioning and assigns a drive letter. Finally, given the virtual disk a name and format it. The new virtual hard disk is accessed through their drive letters.

Involved are hard disks in Disk Management on the menu item *Action*, *virtual hard disk to attach* virtual. Alternatively, this is the tool *VHD Attach* (<http://www.jmedved.com>) possible. The size of a virtual disk can be retrofitted with the *VHD Resizer* (<http://vmtoolkit.com>) adapt.

Other

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware. Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems. Requires special features, such as live migration, emulation of other processors, API emulation, injections of hardware faults, debugging and manual settings of the system time is QEMU / KVM (additional) use.

<<< | # # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_Microsoft_Virtual_PC"

This page has been accessed 10,220 times. This page was last updated on 27 September 2010 at 06:55 clock changed. Content is available under GNU Free Documentation License 1.2 .

VMware ESXi Hypervisor VMware VirtualCenter Server HOWTO vmware, tools, libvirt

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and Emulators / VMware ESXi hypervisor]])

<<< | # # # | >>> | English

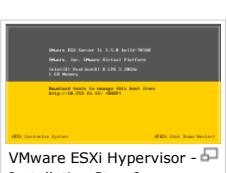
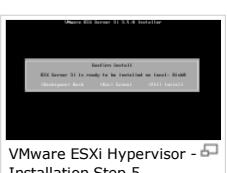
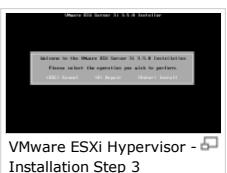
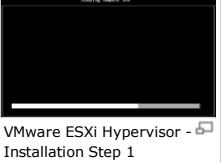
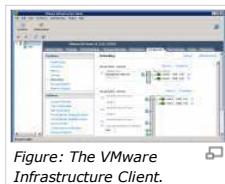


Figure: The web interface of the ESXi server.

Contents	
1 VMware ESXi Hypervisor	
1.1 Installation	
1.2 Configuration using VMware Infrastructure Client	
1.3 Management Tools	
1.3.1 VMware Infrastructure Client (Windows)	
1.3.2 Remote Command Line Interface (RCLI)	
1.3.3 VMware Infrastructure Management Assistant (VIMA)	
1.3.4 VMware VirtualCenter Server 2.5	
1.3.5 The library libvirt	
4.1 VMware Tools	
1.5 Other	

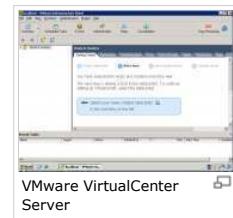


VMware ESXi Hypervisor

Download: <http://www.vmware.com/products/esxi/> (after registration)

The free VMware ESXi server is a type 1 hypervisor and requires no host operating system (bare metal). VMware Server supports multi-processor systems with AMD64 or Intel64. recommend is ESXi servers on VMware certified hardware to run the VMWare (see http://www.vmware.com/pdf/vi35_systems_guide.pdf). The main application field of the VMware ESXi server is the server consolidation. VMware ESXi Server is the first software layer on the hardware. The controller takes a specially adapted Linux, which runs as a virtual machine with specific rights to the hypervisor. On a separate partition with a specific file system, the VMFS, the virtual machine are stored. In a further partition (VMcore) on error core dump information is stored.

The VMWare ESXi has, in contrast to the VMware ESX server, no service console. By removing the service console has indeed the memory requirements of the VMware ESXi hypervisor is reduced to less than 32 MB, many console-based management functions (monitoring, backup, update) are not possible. These management tasks can be supported by the free Remote Command Line Interface (RCLI) (see below). Convenient to management tasks via GUI with the free *VMware Infrastructure Client* handles the. The best management tool is the VMware Virtual Center Server.



Installation

First we need to ensure that support for hardware virtualization in the BIOS of the computer is turned on. The *Demand Based Power Management* is deactivated. It is at least one SATA, SAS or SCSI hard drive to the host and guest systems necessary. It is better to take the guest systems on iSCSI or NFS server. The best solution is a SAN connection with optical fibers. For a test installation, a GB of RAM is sufficient. For production use, it should be much more. There are at least two network interface cards required. One for the management and the second for the network of host systems. The management network should be separated with a router (firewall) from the other networks. NTP and Syslog should only be allowed through. For SAN connectivity via iSCSI or NFS an additional network card is installed. In all the network cards PXE should be disconnected.

The image of the installation CD is to download and burn to a CD. Before you install the security, all network cables are removed. It is the machine with the installation CD to boot. After selecting *installation* by pressing the [Enter] and confirm the license agreement with [F11] to select the partition. Then the installation by pressing the [F11] will start.

After installation and reboot is necessary, the basic configuration in the *Direct Console User Interface* (DCUI). So, press [F2] is to push. The password for the *root* user is set to *Configure Root Password*. It will put the network cable for the management network again. For the management network interface's IP address, netmask, and gateway is defined:

```
Configure Management Network
Network Adapters
IP Configuration
Set static IP address and network configuration
```

Under *DNS Configuration* Server, the IP addresses of the DNS entered. For *Custom DNS suffix* of the DNS suffix is defined. Then in the main menu, select *Restart Network Management*. Now, all network cables are connected.

The SSH access is disabled by default. From ESXi 4.1, steps are necessary:

```
[F2] Customize System
Touble shooting Options
Enable Remote Tech Support (SSH)
```

In older versions of ESXi following steps are necessary: If the *state* after the boot screen appears, is the keyboard combination [ALT] + [F1] key to. In the opening shell environment is the word entered *unsupported* and press [Enter] to confirm with. The entered characters appear not on the screen. Then open the shell and you have root privileges.

```
Tech Support Mode successfully accessed.
The time and date of this access have been sent to the system logs.
WARNING - Tech Support Mode is not supported unless used in
consultation with VMware tech support.
```

To permanently enable SSH access to, *inetd.conf*, the file / etc / editing. The comment character # in front of this line is removed.

```
ssh stream tcp nowait root / sbin / dropbear dropbear multi ...
```

With an SSH client log into from another computer on the Dom0 the ESXi server. VMware ESXi Hypervisor has, in contrast to the VMware ESX server, no service console. Therefore, hardly any administrative tasks possible. You can manually set the configuration files / vmware / etc are secured / under. The actual configuration file is / etc / vmware / esx.conf. For the web interface should SSL certificates are used. Self-signed certificates with the certtool or the wrapper / usr / lib / ssl / misc / l ca.pem on a Linux machine can be generated. These are the ESCI-server copy with scp (see also http://www.vm-help.com/esx/esx3i/change_name_and_cert.php).

```
server.key myesx scp: // host ssl_key
server.pem myesx scp: // host ssl_cert
```

The other configuration is done by a Microsoft Windows machine with the *VMware Infrastructure Client* (GUI) or by *remote command line interface* (command line).

Configuration with VMware Infrastructure Client

To install the *VMware Infrastructure Client* connects you with a web browser via HTTPS with the Web interface to the IP address of the ESXi server. There you invite, download *VMware Infrastructure Client* installation program downloaded this and installed it. After installation, you start the *VMware Infrastructure Client* and connect to the ESXi server. To configure, click on the icon changes to the *Inventory* and *Configuration* tab.

The NICs and virtual switches for the host systems are configured as follows:

```
Configuration> Networking> Add Networking> Virtual Machine> Create a virtual switch network Label = Production VLAN ID = Empty Configuration> Networki...
```

The license key is entered.

```
Configuration>
Licensed Features>
License Source>
Edit>
Use Serial Number
```

The NTP server is specified.

```
Configuration>
Time Configuration>
Properties>
Options
NTP Settings
```

```
Add> esxgw.mycompany.com
General> Start automatically
```

The Syslog server is configured as follows.

```
Configuration>
Advanced Settings>
Syslog>
  Remote
    Syslog.Remote.Hostname = syslogserver.meinedomain.com
    Syslog.Remote.Port = 514
```

For SFCB resource limits can be set.

```
Configuration>
System Resource Allocation>
Advanced>
  host>
    vim>
      sfcbs>
        Edit Settings>
          CPU Resources> Limit = 500 MHz
```

At system startup, the virtual machines are started automatically. Before shutting down the host system should include the guest systems are shut down.

```
Configuration>
Virtual Machine Startup / Shutdown>
Properties>
  Allow virtual machines to start and stop automatically
  with the system
```

If an iSCSI server on the network available at the address indicated.

```
Configuration> Storage Adapters> iSCSI Software Adapter> General Properties> Configure> Enabled Dynamic Discovery> Add> iSCSI Server = xxx.xxx.xxx.xxx
```

If an NFS server on the network available at the address indicated.

```
Configuration> Storage> Network File System
Server = xxx.xxx.xxx.xxx
Folder = / some / path
Datastore name = xxx
```

Management Tools

For remote administration of the ESXi server *client*, among the already mentioned *VMware Infrastructure* (Microsoft Windows), the free *Remote Command Line Interface* (Linux, Microsoft Windows) and the paid *VMware Virtual Center Server* (Microsoft Windows) is available.

VMware Infrastructure Client (Windows)

Download: Web interface of the ESXi server.

The configuration described in the *VMware Infrastructure Client* is already available for Microsoft Windows. It is used on other computers for remote administration. First you need to connect to the host on which the VMware Server runs on. The virtual machines seem to manage in a list. The creation of virtual machines is done with a wizard.

Remote Command Line Interface (RCLI)

Download: <http://www.vmware.com/go/remotedcli>

RCLI stands for Linux and Microsoft Windows. Furthermore, a *virtual appliance* is available with RCLI. This virtual machine can be operated on the ESXi server. In the download URL RCLI the documentation can be downloaded. To install the *Remote Command Line Interface* on Linux, the following steps are necessary. For 64-bit operating system version is to select the *x86_64*.

```
~ $ Tar xzf VMware-RCLI-*.tar.gz-C / tmp
~ $ Cd / tmp / vmware-distrib-RCLI
~ $ Sudo . / Vmware-install.pl
```

Alternatively, the *appliance* used. After downloading the ZIP file is unpacked and Microsoft Windows machines with the *VMware Infrastructure Client* to copy the. In the *VMware Infrastructure Client Appliance* is the *menu File*, *Virtual Appliance*, *Import* to import the above. In the Wizard is to *import from File* to select the OVF file. After the virtual machine list is published in, it is a right-click and *power on* on the shortcut menu started with. The tab *Console* log into the system in. This is the root password set. The network configuration via DHCP. For a fixed IP address *interfaces* is the file / *etc / network / adapt*. This is a *root* user can log in from outside and is in the file / *etc / ssh / sshd_config* *PermitRootLogin yes* to set the setting. The *reboot* command changes makes them effectively.

To install the Microsoft Windows RCLI exe is the file *VMware-*.VIRremoteCLI* download and execute. To implement RCLI commands in the *bin* subdirectory of the installation path to change.

After each installation, the RCLI commands are used. Here is a small selection. The command *resxtop* shows, similar to the Unix *top* command, use the resources of a ESXi or ESX server. *The-h* option lists the options. The following will use the resources of the server *myesx*.

```
~ $ Resxtop - server myesx - username root
```

The option *-server* is unnecessary if the variable was defined *VI_SERVER*.

```
~ $ Export = VI_SERVER myesxi
```

With the following command RCLI name is a backup of the configuration on the server with the ESCI *myesx* created.

```
~ $ Vcfg-cfgbackup-s - server myesx - username root myesx.bak
```

RCLI the command *vmware-cmd* with virtual machines are managed (start, stop, suspend, snapshot). The following command lists all registered virtual machines.

```
~ $ Vmware-cmd - server myesx - username root-1
```

By RCLI command *svmotion* virtual machines can be during the ongoing operation of a ESXi server to another move. For this purpose, a common storage of the ESXi server is necessary.

VMware Infrastructure Management Assistant (VIMA)

Website: <http://www.vmware.com/support/developer/vima/>

The *VMware Infrastructure Management Assistant* (VIMA) is a virtual machine that allows administrators and developers, and systems to manage ESXi ESX. With the *VI Client Appliance*, this *menu File*, *Virtual Appliance*, *importers* to import tons of it. In the Wizard is to *import from URL* with <http://www.vmware.com/go/importvima/vima1.ovf> indicated. After the virtual machine list is published in, it is a right-click and *power on* on the shortcut menu started with. The tab *Console* log into the system in. Previously, *admin* is the network configuration and the password for the user to specify *vi*. The documentation to load the URL <http://www.vmware.com/support/developer/vima/vima10/doc/vima10guide.pdf> down.

VMware VirtualCenter Server 2.5

Download: <http://www.vmware.com/de/download/> (time-limited evaluation version)

The *VMware VirtualCenter Server* is a powerful management software for ESX and ESXi. To test you download a time limited evaluation version. Installation can be done on a virtual machine. The operating system is Windows 2003 Server Standard Edition to choose. The IIS Web server may not be installed. The Windows server locale is *English (USA)* to adjust to. This is done in the Control Panel from the Settings for the language.

To install the *VMware VirtualCenter* is the installation file to download and unpack. In the applied part is the *autorun.exe* file to start. First is the installation language to rely on *English*. This is followed by your user name and organization. Then is to click *Next* again. It is *VMware VirtualCenter Server* and the next step *Install MS-SQL2005 Express* select. *Evaluate the test server* is activated. The following inputs necessary (example):

```
VC Server IP: myvmvc.mydomain.com
Administrator: Administrator
Password: *****
```

With *Next*, *Install*, *Finish*, the installation is complete. *Client* is called the *VMware VirtualCenter Server* from the *Start menu*, *All program*, *VMware*, *VMware Infrastructure*. First you have to login to the computer on which the *VMware VirtualCenter Server* is running. The first configuration, a data center must be created. The data center must be assigned one or more hosts. In this example it is the *ESXi* server. Then virtual machines and templates can be created. The *VMware VirtualCenter Server* also has a web interface for administration of virtual machines. At the point *Web access log* under the URL <https://myvmvc.mydomain.com> (example) is logged on.

The libvirt library

Website: <http://libvirt.org>

libvirt, a virtualization layer between software and management tools. Thus, the development of management tools for all virtualization solutions that support this library possible. The *libvirt* library supports QEMU, KVM, VirtualBox, VMware ESX, Xen, LXC Linux container system, OpenVZ Linux container system and User Mode Linux *libvirt*. Is published as open source. For various Linux distributions, corresponding software packages are installed. The library can be compiled with Cygwin and Mingw under Microsoft Windows. The installation and use is at the URL http://qemu-buch.de/d/Managementtools/_libvirt-Tools described. A command reference can be found at the URL http://qemu-buch.de/d/Anhang/_libvirt. The following example connects you with *virsh* to the host *my-esx-server* and allows the current machine list.

```
~ $ Virsh
virsh # connect esx: / / my-esx-server / 1? no_verify =
virsh # list - all
 Id Name Status
-----
272 webserver01 continuously
288 webserver02 continuously
304 windowsworkstation01 continuously
384 print server running
480 nagios (monitoring) continuously
624 honeypot01 continuously
ReactOS 672 (test) running
 - Turn off developer01
 - Turn off honeypot02
```

VMware Tools

It is recommended to the guest systems, the *VMware Tools* install. They optimize with special drivers to work with the virtual machine. It can be the resolution of the host window to adjust it and it will *cut & paste*. The *VMware tools* for the different host systems as ISO images (*freebsd.iso*, *linux.iso*, *netware.iso*, *windows.iso*) and have offered to install in the guest system are involved.

For Unix like host systems (*VMware* is to monitor the tools with *Monit* <http://mmonit.com/monit/>) is recommended. starts *Monit* stopped *VMware Tools*. The installation of the package *monit* Ubuntu using a command line does under.

```
Guest ~ $ sudo apt-get install monit
```

is then in */etc/default/monit* *startup = 0* to the line *1 = Change startup*. *Monit* is configured with the configuration file */etc/monit/monitrc*.

```
# / Etc / monit / monitrc
# Email Alerts for
set mail-format {
    from: root@mygast.mydomain.de
    subject: [$ ACTION] $ HOST $ SERVICE $ EVENT - monit
}
set alert my-email@mydomain.de
# # VMware-tools
check process vmware-tools with pidfile / var / run / vmtoolsd.pid
my-email@mydomain.de alert only on nonexistent {}
start program "/ etc / init.d / vmware-tools start"
stop program "/ etc / init.d / vmware-tools stop"
if 5 restarts within 5 cycles then timeout
```

It is to test the configuration and restart *Monit*.

```
Guest ~ $ / etc / init.d / monit syntax
Control file syntax OK
Guest ~ $ / etc / init.d / monit
```

Other

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware. Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems. Requires special features, such as live migration, emulation of other processors, API emulation, injections of hardware faults, debugging and manual settings of the system time is QEMU / KVM (additional) use.

<<< | # # | >>>

Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_VMware_ESXi_Hypervisor"

This page has been accessed 37,066 times. This page was last updated on 29 September 2010 at 08:26 clock changed. Content is available under GNU Free Documentation License 1.2 .

VMware Player 3, vmware-tools, VMware Converter vCenter, MokaFive

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and Emulators / VMware Player]])

<<< | # # # | >>> | English

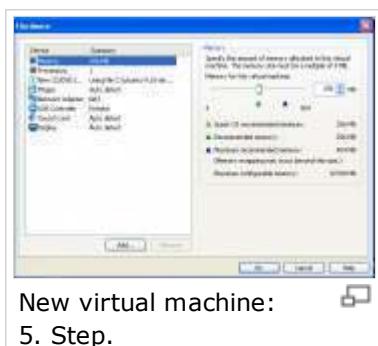


The VMware Player.

New virtual machine:
1. Step.New virtual machine:
2. Step.New virtual machine:
3. Step.



New virtual machine:
4. Step.



New virtual machine:
5. Step.



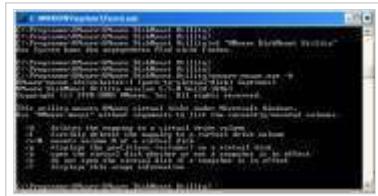
New virtual machine:
6. Step.



The VMware Player virtual
machines.



The BIOS of VMware Player.



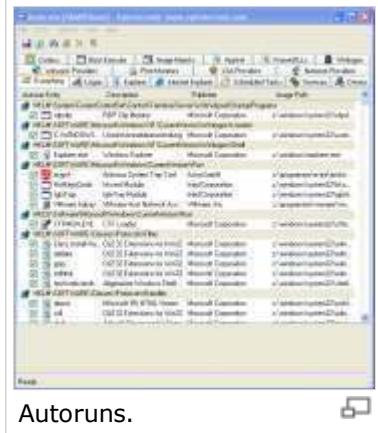
VMware Workstation 5.5 Disk Mount Utility.



VMware disk mount GUI.



CCleaner cleans Microsoft Windows.



Autoruns.



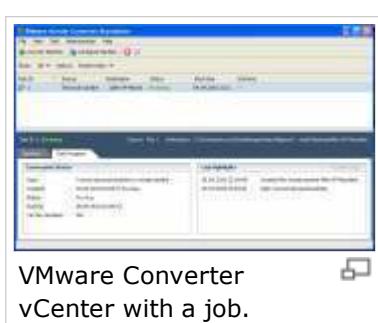
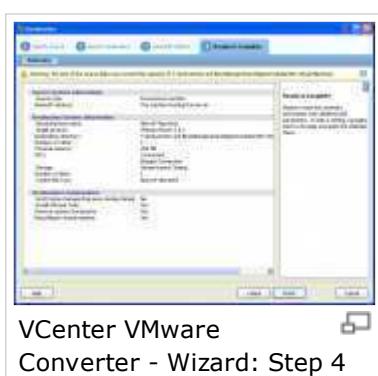
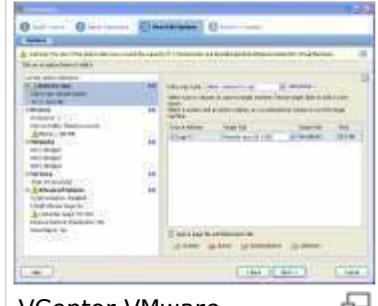
VCenter VMware Converter.



VMware vCenter Converter - Wizard: Step 1



VCenter VMware Converter - Wizard: Step 2



Contents

- 1 VMware Player 3**
 - 1.1 Installation
 - 1.1.1 Microsoft Windows
 - Linux 1.1.2
 - 2.1 Creating Virtual Machines
 - 1.3 Virtual Appliances
 - 1.4 virtual machines and emulators of other virtualizer
 - 1.5 XP-Mode Import
 - 1.6 VMware Tools
 - 1.6.1 Windows-guest systems
 - 1.6.2 Linux guest systems
 - 1.7 control of the running virtual machine
 - 1.7.1 Full Screen
 - Unity 1.7.2
 - 1.7.3 Reset, Suspend, and Power Off
 - 1.7.4 Snapshots?
 - 1.8 Hardware configuration of the virtual machine
 - 1.8.1 Memory
 - 1.8.2 Processor
 - 1.8.3 Hard Disk
 - 1.8.4 CD / DVD
 - 1.8.5 Floppy
 - 1.8.6 Network Adapter
 - 1.8.7 USB Controller
 - 1.8.8 Sound Card
 - 1.8.9 Display
 - 1.9 Other configurations of the virtual machine
 - 1.9.1 General
 - 1.9.2 Power
 - 1.9.3 Shared Folder
 - 1.9.4 Tools
 - Unity 1.9.5
 - 1.10 The virtual BIOS
- 2 Additional Software**
 - 2.1 VMware Disk Mount Utility VMXBuilder
 - VMware Converter 2.2 vCenter
- 3 Moka Five Players**
- 4 Other**



The Moka Five Player USB - Settings.

VMware Player 3

Download: <http://www.vmware.com/products/player/> (after registration)

VMware Player (from VMware Inc.) is based on the native virtualization, and supports full virtualization. It enables on computers with x86 or x86-64 processors, multiple virtual machines for these processors to use. VMware Player runs on Linux and Windows. The free VMware Player provides the ability to virtual machines created with VMware Workstation, VMware Server or VMware ESX Server to run. The VMware Player is Open Virtualization Format (OVF) and the Open Virtual Appliance (OVA). The VMware Player is *Microsoft Virtual PC* and *Microsoft Virtual Server* and *Virtual Open* operate machinery. VMware Player also supports the image of *Symantec Backup Exec System Recovery* (formerly *LiveState*). From VMware Player 3, you can move virtual machines. The installation of supported guest systems is comfortably possible with *Easy Install*. On Microsoft Windows 7, the VMware Player to import the XP mode. VMware Player supports the Aero desktop, Microsoft Windows 7 Using the technique *Thin Print* to print the host system directly to the printer connected to the host system. There are no snapshots, like QEMU / KVM possible.

Installation

Microsoft Windows

After downloading, you start the *vmware-player-*.exe*. It is thereby the target directory *C:\Program Files VMware\VMware Player\preset*. With *Change* this setting can be changed. Then asked if *shortcuts* are created to call for the comfortable. These settings do not change. At the end of the installation Microsoft Windows is restarted. Is called the VMware Player from the *Start menu, Programs, VMware, VMware Player*. After the first start the license agreement is to be confirmed.

To import the Microsoft Windows CD and DVD as an image, additional software is required. Most CD / DVD burning programs support importing from CD / DVDs. A free program is ImgBurn (<http://www.imgburn.com>). It loads the installer *SetupImgBurn_*.exe* from the site now and start it. After starting the program, choose *Create image file from disc* to ISO image of the install disc to create a.

Linux

Before installing these packages are installed (eg Ubuntu 9.04).

```
Host ~ $ sudo apt-get install build-essential linux-headers-`uname -r`
```

It is the latest version of VMware Player (32 - or 64-bit) download. The graphical installer is started as follows.

```
Host ~ $ chmod +x ./VMware-Player-2.5.1-126130.i386.bundle
Host ~ $ gksudo bash ./VMware-Player-2.5.1-126130.i386.bundle
```

Installation under Linux is self-explanatory. It installs special kernel modules. Importing from CD / DVDs is at the URL http://qemu-buch.de/d/Speichermedien/_Images_anlegen described.

Creating virtual machines

In VMware Player *Create a New Virtual Machine* Wizard to create a new virtual machine is started with. The first step is to select the installation media. With *Installer disc*, a DVD in the host system inserted CD / used. In this example, the installation of Ubuntu to an ISO image. The ISO file to *installer disc image file* to select. For supported host systems using the VMware Player *Install* a technique called *Easy*. This installation is largely automatic. *Easy Install* on Ubuntu asks at the beginning only after the user to be created. The next step in the name of the virtual machine and the target directory is specified. Subsequently, the size of the virtual disk is to be specified. Larger boards are better to invest in several parts. In the overview at the end of the wizard makes the *hardware Customize* with virtual hardware to adapt. With the *Finish* button to start the installation of the host system in the virtual machine. The demand for the installation of VMware Tools must be answered positively. In order to have control of the host system, you have to click in the window of the virtual machine. To return to the desktop of the host system, the key combination [Ctrl] + is [Alt] necessary.

If a virtual machine is often used to set up a quick start shortcut. To do this click the right mouse button on the Windows desktop and choose *New, Shortcut*. *Browse* to choose VMX file of the desired virtual machine from the. These are usually in *My Documents\My Virtual Machines*. Subsequently, this linkage should be designated. With a double click on the link you start the virtual machine.

Virtual Appliances

Fully installed virtual machines (virtual appliances), drawn by the URL <http://www.vmware.com/appliances/> download. This site will open when the menu item *File, Download a Virtual Appliance* enabled. You look either in the categories, for example, *operating system*, or using the search box. After downloading the archive is unpacked. About the *File menu, open a virtual machine* is the appropriate VMware configuration file (*.vmx to open).

Virtual machines and emulators of other virtualizer

Will virtual machines from *Microsoft Virtual PC* or *Microsoft Virtual Server*, open the VMware Player generates a new VMX file. The original VMC file remains unchanged. Images are *Symantec Backup Exec System Recovery* open, generates the VMware Player, a new VMX file. The original image (.Sv2i) remains unchanged. With *qemu-img* (see http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware) and *ubuntu-vm-builder* (see http://qemu-buch.de/d/Managementtools/_libvirt-Tools) Virtual machines can also be created and converted. Microsoft

Windows as guest system detects at boot for new hardware and install default drivers. Then a restart of the host system is necessary.

XP-Mode Import

The XP-mode Microsoft Windows 7 Professional, Ultimate, and Enterprise is a free image with a custom Windows XP Professional. This virtual machine is normally powered by Microsoft Virtual PC 2007. After installing the VMware Player is the URL of the XP-Mode <http://www.microsoft.com/windows/virtual-pc/download.aspx> download. Then file is the XP mode by calling the *us.exe* to install *WindowsXPMode_de*. After installing XP-Mode in VMware Player is imported. This is done via the menu *File, Import Fashion Windows XP VM*. The VMware Player can find the XP mode and rename the new virtual machine with *Windows XP mode*. After starting the virtual machine, the VMware Tools are installed.

VMware Tools

If not done, the VMware Tools are installed. This is a CD image with various drivers. This can be set as higher screen resolutions. A common buffer is used. Another advantage is the easy exchange of the mouse pointer between the desktop and the virtual machine window. The installation is the running virtual machine via the menu option *VM, install VMware Tools ...* introduced in. Here, the virtual CD to the virtual CD drive on the host system is loaded. The CD image with VMware Tools is loaded on first use of the VMware Web site. More information can be found at the URL <http://www.vmware.com/support/pubs/>.

Windows guest systems

Microsoft Windows is usually the case when you insert the virtual CD automatically called in the file *setup.exe*. If it fails, the automatic start, so manually in the file manager the file *setup.exe* to start. It is typical to select the installation type. Then, the host system is restarted. To set a higher screen resolution in the guest system, just click the right mouse button on any free space on the Windows desktop. Under the *Settings* tab to configure the screen resolution.

Linux guest systems

On Unix / Linux as host system, this CD-ROM is included. This is normally done automatically. The tarball is unpacked and the script *vmware-install.pl* call. They are the kernel headers and development tools necessary. On Ubuntu, the following commands are entered. After each kernel update, the script re-run *vmware-install.pl*.

```
Guest ~ $ sudo cp / * media/cdrom1/VMwareTools.
Guest ~ $ sudo umount / dev / cdrom
Guest ~ $ tar xzvf *. tar.gz VMwareTools
Guest ~ $ cd vmware-tools-distrib
Guest ~ $ sudo . / Vmware-install.pl -d
```

Control of the running virtual machine

Full screen

This completes the virtual machine window the entire screen, the upper window frame is to click twice. With a small bar at the top of the screen can be controlled by the virtual machine.

Unity

The Unity feature lets you merge the desktop of the host system to the desktop of the host system. The applications of the host system running as a window on the host system. The Unity feature is available after installing the VMware tools. *Unity* in the current virtual machine, the *VM menu, Enter Unity* activated. The start menu of the host system is activated via the key combination [Ctrl] + [Shift] + [V].

Reset, Suspend, and Power Off

With the menu-point *VM, Power*, the commands are *reset*, *suspend* and *power off* available. *Reset* and *Power Off* installed VMware tools should be used only at, or threatened loss of data. In *Suspend* the state of the virtual machine is stored before it is finished. The next time you start this saved state is restored.

Snapshots?

The VMware Player 3 does not support snapshots. I need a backup point, we save the state of the virtual machine with instant-messaging *VM, Power, Suspend*. Then you copy the directory of the virtual machine. On Microsoft Windows it is usually located in *My Documents \ My Virtual Machines*. If necessary, you open the copied virtual machine.

Hardware configuration of the virtual machine

To configure the virtual machine must be shut down. The corresponding dialog box you call the menu *VM, Settings*. Under the *Hardware* tab, the virtual hardware is configured. New hardware is added using the *Add* button. It helps a wizard. Supported Linux and Windows guest systems can *print* directly to printers with *Thin* the host system to print on. *Add* to this is to use *Virtual Printer*. The guest system configures the printer automatically.

Memory

The amount of memory (RAM) of the virtual machine is configured here.

Processor

It is the number of emulated CPUs pretend (SMP). The settings under *Virtualization engine* usually do not need to be changed. For special guest systems and test environments, the type of virtualization / emulation is adapted.

- *Automatic*
- *Automatic with replay*
- *Binary translation*
- *Intel VT-x or AMD-V*: This option is only available if the host system that supports full virtualization.
- *Interl-VT-x/EPT or AMD-V/RVI*: This option is only available if the host system that supports full virtualization.

Hard Disk

Among *utilities*, the following functions:

- *Map ...*: Adds the virtual hard drive as a drive in the host system. The host system must adjust the file system that drive can read. It is recommended to read to use the virtual disk (*Open file in read-only mode*).
- *Defragment*: Virtual Hard Drive Microsoft Windows guest systems can be defragmented. On Linux this is not necessary.
- *... Expand*: Enlarges the virtual hard disk. In the host system on the additional space, additional partitions are created and formatted.
- *Compact*: Optimizes a virtual hard disk.

CD / DVD

An ISO image or CD / DVD drive of the host-guest system can be made available to the system. *Connect at power on* is activated.

Floppy

An image or the floppy drive of the host-guest system can be made available to the system. *Connect at power on* is activated.

Network Adapter

The following types of networks to choose from.

- *Bridged*: The virtual machine is assigned a unique IP address on the network. The IP address it receives from either a DHCP server on the network, or directly from the Internet provider. *Bridged* is most often used.
- *NAT*: Network Address Translation is at the internal IP address of the host system to the outside with the IP address of the host system to map the. The internal DHCP server assigns to the host

system an IP address from 192.168.81.0. Also, the routing through the provided gateway is defined. A network configuration for most operating systems is not necessary and access from the host system to the outside is now available. The firewall blocks all incoming instance of connections to this virtual network. Therefore it can not be accessed from outside the virtual machine. The host system is protected.

- **Host Only:** It is private network to the host system formed a. Outside the host system, this network is not visible. The internal DHCP server assigns the guest system automatically assigns an IP address.

After installing the VMware Player VMware two new network cards are set up. On Linux, the *ifconfig* command provides information about these adapters. On Microsoft Windows Control Panel is to call the (*Start Menu, Control Panel*). Under *System, Hardware tab* you open the *Device Manager*. Under *Network Adapter VMnet8* the two VMware Network Adapter *VMnet1* and listed.

USB Controller

The following options are available:

- *Enable high-speed support for USB 2.0 devices:* This option should be enabled.
- *Automatically connect new USB devices:* new USB devices connected to the host system system are automatically uploaded to the guest.
- *Show all USB input devices:* There are all USB devices to the host system of the host system shown.

Sound Card

The sound output can be disabled or set a different sound card.

Display

It can enable the 3D acceleration (*Accelerate 3D graphics*). Furthermore, the number of monitors and their resolution will be set.

Other configurations of the virtual machine

To configure the virtual machine must be shut down. The corresponding dialog box you call the menu *VM, Settings*. Under the *Options tab*, the following functions:

General

Here, the name of the virtual machine, operating system and the default location to be changed.

Power

The automatic full screen mode after the start is activated. Furthermore, the host system the battery status is transmitted.

Shared Folder

To exchange data between the guest and host system are *shared folder* (shared folders) uses. There must have installed the VMware Tools. To set up a *Shared Folder* is *Always* select *enabled*. With the *Add button* system, a path to the host selected. The shared folder is read-only system (*read-only*) assigned to the guest. In Microsoft Windows Explorer as a guest system is started and *Shared Folders* to open the entry.

Tools

The upgrade of VMware tools is defined.

Unity

The behavior when *Unity* is adapted here. In addition to the configuration of the window frame, the menu of the applications (de-) activated.

The virtual BIOS

Each virtual machine under VMware products have like a real PC BIOS (Basic Input Output System). The

BIOS enables communication between the operating system and hardware. When booting the virtual machine can be reached by pressing the [F2] in the virtual BIOS. If you want to use often live CD / DVD, one is under the *boot CD-ROM drive* first. This is using the cursor keys to select *CD-ROM drive* and press the] several times to press + [.

Additional Software

VMware Disk Mount Utility VMXBuilder

Downloads:

- http://www.vmware.com/download/eula/diskmount_ws_v55.html (VMware Disk Mount Utility)
- <http://www.vmxbuilder.com> (VMXBuilder, VMware disk mount GUI)

With the free *VMware Workstation 5.5 Disk Mount Utility* can be a virtual disk on the host system (Microsoft Windows) embed. Given the virtual machine must be off. Needs to continue to the host system's file system can read the virtual machine. After downloading the file *VMware-mount-*.exe* that is run. Installation is done with a wizard. The target directory is by default *C:\Program Files\VMware\VMware disk mount utility*. After installation of obtaining information about the options to this command-line tools.

```
Host C: \> cd C: \ Program Files \ VMware \ VMware disk mount utility \
Host C: \> vmware-mount.exe-h
```

A graphical interface for that program enables the *VMware disk mount GUI*, the *VMXBuilder* is contained in the package. After downloading the installation file to start it. A wizard helps with the installation. Is called the GUI on the *Start menu, Programs, Devfarm software VMXBuilder, VMware disk mount GUI*. Desktop *Drive Letter* drive letter is free to choose. Desktop *Virtual Disk File* is chosen with the icon [...] the path of the virtual hard drive to. To integrate the virtual disk, click on *Mount*. In Windows Explorer to access the virtual hard disk on the specified drive letter is possible. To release you switch to the tab *Unmount Virtual Disk*. There one chooses behind *Drive Letter* from the appropriate drive letter and click *Unmount*. This may be no access to this drive.

vCenter VMware Converter

Download: <http://www.vmware.com/products/converter/>

With the free *VMware Converter* can vCenter existing physical computer with Microsoft Windows NT4 SP4 +, 2000, XP, Server 2003, Vista and 7 virtual machines convert. Furthermore, the following Linux distributions are supported: Red Hat Enterprise, Red Hat Advanced Server, SUSE Linux Enterprise Server and Ubuntu. The transfer of an operating system from one physical machine to a virtual machine is called physical to virtual (see http://qemu-buch.de/d/Speichermedien/_Physical-to-Virtual). For a running operating system runs with applications and data without reinstalling old hardware into a virtual machine. *VMware vCenter Converter* also allows you to move from host systems, another virtualization solution, such as Virtual PC, in a VMware product. The conversion of virtual machines is also referred to as a V2V migration (see http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware). The steps for moving a system are the following: It is the *VMware Converter vCenter* installed. To convert the system to be optimized. This includes the deletion of unnecessary garbage and cleaning the Windows registry. Furthermore, all startup entries are to be deactivated.

After this brief review now for the precise description of the steps. After downloading the installation file *vmware-converter-*.exe* it starts. In the Installation Wizard, the default destination directory is assumed. In the *Setup Type* dialog to take the pre-selected *local installation*. Before the *VMware vCenter Converter* is started, the first system to optimize.

The elimination of garbage can with the free program *CCleaner* (http://www.filehippo.com/download_ccleaner_done). After installing and launching this program, click on *Analyze*. Then enable *Run Cleaner* in order to dispose of the junk file found. To clean the Registry, the *Registry* button and select *Check for errors* to click. After the analysis is activated to *fix mistakes!*

Thus, the virtual Windows startup no error messages produced are cut off prior to the move, the startup entries. After the move they have to activate again. Useful (this is the free program *Autorun*

<http://technet.microsoft.com/de-de/sysinternals/bb963902.aspx>). After installing and starting the program, choose the tab *autoruns.exe Everything*. To disable the entries are listed.

The *VMware Converter* is *vCenter* the *Start menu, Programs, VMware Converter standalone client* called about. The wizard to convert *Convert Machine* is started. In the first step (*Specify Source*) is chosen behind the *Select source type on point-powered machine*. Furthermore, *This* is to enable *local machine*. The next step (*Specify Destination*) is *Select destination type*, the point behind *VMware Workstation VMware or other virtual machine* selected. For *Select VMware* to enable *VMware Player 2.5.x product*. It is the name and path for the virtual machine to pretend. In this example, the removal of the host operating system is in a virtual machine on the same computer. To do this, according to the space available. This can be ensured by an additional (USB) hard drive or network drive.

In step *View / Edit Options* settings to adjust so that all the yellow warnings disappear. Each case it is the right link to click on the *Edit*. *Desktop Data* to choose *copy* to copy the hard drive partitions. It is *file Ignore page file and hibernation* activate. In *memory devices*, the number of CPUs and the size of the match. *Networks* under network cards, the number of set. It should be noted that the target system does not receive the same IP address as the current source system. In *services*, the services to be started set. Critical services, such as virus scanners and firewall should be disabled for the time being. Otherwise, there are problems starting the virtual machine. Under *Advanced options* enable *Install VMware Tools on the imported virtual machine*.

In the last step (*Ready for Complete*) we checked again the specifications and click *Finish* to start the parade. The conversion is complete, you open the VMware Player, the VMX file of the new virtual machine from the *File menu, open a virtual machine*.

Moka Five Players

Website: <http://www.mokafive.com>

Download: http://www.freewaregeeks.com/?page=detail&get_id=2073&category=66

The Moka Five Player is based on the VMware Player. With him it's just virtual machines ready to download from the Internet ([## Other](http://lab.mokafive.com>List). The variant Moka Five Player USB allows you to start virtual machines from a USB flash drive. Unfortunately, the company's Web site Moka Five a download link for the Moka Five Player.</p>
</div>
<div data-bbox=)

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware. Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems. More features than the offer of the free VMware Player VMware Server (see http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_VMware-Server) and VMware Workstation (see http://qemu-buch.de/d/Notes/_Weitere_Virtualisierer_und_Emulator/_VMware_workstation). Instead of fee-based VMware Workstation can use VirtualBox itself (see http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_VirtualBox). Requires special features, such as live migration, emulation of other processors, API emulation, injections of hardware faults, debugging and manual settings of the system time is QEMU / KVM (additional) use.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulator/_VMware-Player "

This page has been accessed 9565 times. This page was last updated on 27 September 2010 at 06:56 clock changed. Content is available under GNU Free Documentation License 1.2 .

VMware Server 2 vmrun, libvirt

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and Emulators / VMware Server]])

<<< | # # # | >>> | English



VMware Server 2
Login Web Interface.



VMware Server 2
Web Interface.



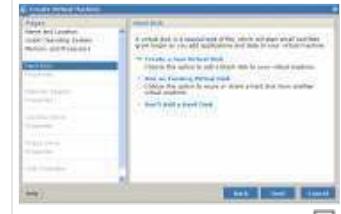
VMware Server 2
Creating a Virtual Machine:
Step 1



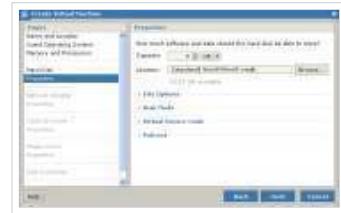
VMware Server 2
Creating a Virtual Machine:
Step 2



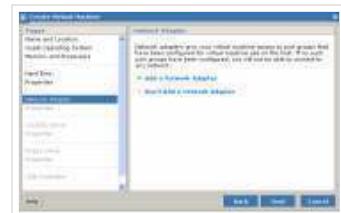
VMware Server 2
Creating a Virtual Machine:
Step 3



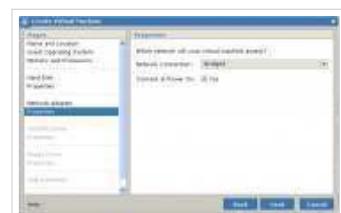
VMware Server 2
Creating a Virtual Machine:
Step 4



VMware Server 2
Creating a Virtual Machine:
Step 5



VMware Server 2
Creating a Virtual Machine:
Step 6



VMware Server 2
Creating a Virtual Machine:
Step 7



VMware Server 2
Creating a Virtual Machine:
Step 8



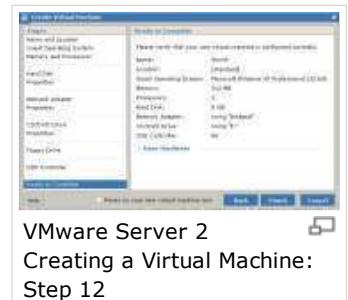
VMware Server 2
Creating a Virtual Machine:
Step 9



VMware Server 2
Creating a Virtual Machine:
Step 10



VMware Server 2
Creating a Virtual Machine:
Step 11



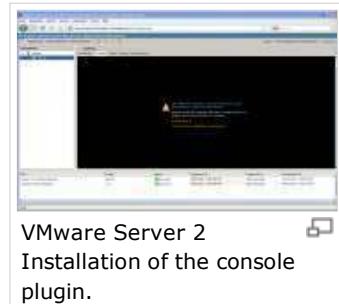
VMware Server 2
Creating a Virtual Machine:
Step 12



VMware Server 2 with a
virtual machine.

Contents

- 1 VMware Server 2.0
 - 1.1 Installation
 - 1.1.1 Microsoft Windows as the host system
 - 1.1.2 Linux host system
 - 1.2 The web interface VI Web Access
 - 1.3 Configuration
 - 4.1 Creating Virtual Machines
 - 1.5 Virtual Appliances
 - 1.6 VMware Tools
 - 1.6.1 Microsoft Windows guest systems
 - 1.6.2 Linux guest systems
 - 1.7 control of the running virtual machine
 - 1.7.1 Full Screen
 - 1.7.2 Power Off, Suspend, Suspend Guest, Guest Shutdown, Reset, and Restart Guest
 - 1.7.3 Access from removable media
 - 1.7.4 Snapshots
 - 1.8 Hardware configuration of the virtual machine
 - 1.8.1 Processor
 - 1.8.2 Memory
 - 1.8.3 Hard Disk
 - 1.8.4 CD / DVD
 - 1.8.5 Network Adapter
 - 1.9 Other configurations of the virtual machine
 - 1.9.1 General
 - 1.9.2 Power
 - 1.9.3 Snapshots
 - 1.9.4 Advanced
 - 1:10 Command Line Tools
 - 1.10.1 vmrun
 - 1.10.2 vmware-vdiskmanager
 - 1.11 More Tools
 - VMware Converter 1.11.1 vCenter
 - 1.11.2 vmbuilder
 - 1.11.3 VMware VirtualCenter Server
 - 1.11.4 openQRM
 - 1.11.5 The libvirt library
 - 1:12 Other



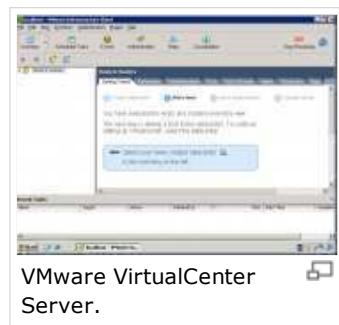
VMware Server 2
Installation of the console
plugin.



VMware Server 2 -
console.



VMware Virtual Network
Editor.



VMware VirtualCenter
Server.

VMware Server 2.0

Download: <http://www.vmware.com/products/server/> (after registration)

The application of the free VMware Server is the server consolidation. Administration of virtual machines is to serve the web interface (*VI Web Access*). The VMware Server is the successor to VMware GSX and is the little brother of VMware ESX (i). The VMware Server, as the Kernel-based Virtual Machine, a type 2 hypervisor and requires a host operating system. VMware Server can be installed on Linux or Microsoft Windows. We recommend a slim Linux distribution such as Ubuntu Server. VMware Server supports full virtualization, but does not depend on them. Therefore the running VMware Server, as opposed to VMWare ESX (i) server, even on modest hardware. This makes it particularly interesting for test and development environments. As a solution for desktop virtualization VMware Server is indeed used but VirtualBox, VMware Player and VMware Workstation are better suited. VMware Server can not be installed with other VMware virtualization products.

With version 2.0 of VMware Server and Microsoft Windows Server 2008 and Microsoft Windows Vista are supported as guest systems. The RAM can be used for each host system may be up to 8 GB of RAM. This is a 64-bit host system or a 32-bit host system with PAE support (Physical Address Extension) is necessary. With CPUs with hardware virtualization, 64-bit guest operating systems. Continues to offer the VMware Server to host systems USB 2.0 and up to ten virtual network interfaces. Virtual SCSI disks assigned to the guest on the fly. Unfortunately, VMware dominates the server only a snapshot per virtual machine.

Installation

To install a freely available license number is required. It can be the URL <http://register.vmware.com/content/registration.html> request multiple license numbers.

Microsoft Windows as the host system

Officially, the VMware Server from Microsoft Windows Server 2000, 2003 and 2008 is supported. Runs is the VMware Server on Microsoft Windows XP and Vista. To install one start the *vmware-server-2.*.exe*. A wizard supports the installation. First, the license terms are acceptable. In the next step) is the installation directory (*Destination Folder* set. The default default *C:\Program Files\VMware\VMware Server* can be accepted. In step *server configuration information*) is the directory for the virtual machine (*virtual machine storage path* adapted. It may be the default setting (*C:\Virtual Machines*) data store to be taken for. Furthermore, the full computer name with domain name (FQDN) is specified. The specifications for *HTTP Server Port* (8222) and *server HTTPS port* (8333) need not be changed. Furthermore *system* is *Allow virtual machines to start and stop with the automatically* activated. In the *Configure* step is to activate all *shortcuts*. At the end of the installation), the registration information (*User Name, Company and Serial Number* to enter.

When you install VMware Server on Microsoft Windows, the error message *File was rejected by digital signature policy* appear. be installed to avoid this error message the Microsoft patch KB925336 (Windows Server 2003). For installation on Microsoft Windows XP registry entry may be a necessary (<http://support.microsoft.com/kb/925336/en-us>). *Autorun*, the CD / DVD automatically starts when inserted action is to disable, It may cause unpredictable problems with the virtual machines. As of version 2.0 of VMware Server on Microsoft Windows for the Web interface to the Apache and the Tomcat server. On the services access this service as *VMware Server Web* filter.

To import the Microsoft Windows CD and DVD as an image, additional software is required. Most CD / DVD burning programs support importing from CD / DVDs. A free program is *ImgBurn* (<http://www.imgburn.com>). It loads the installer *SetupImgBurn_*.exe* from the site now and start it. After starting the program, choose *Create image file from disc* to ISO image of the install disc to create a. The ISO images are to be stored in the data store. On Microsoft Windows this is usually the *C:\Virtual Machines*.

Linux host system

It is recommended to create a separate partition for virtual machines. As a host system including Linux distributions, Ubuntu, Red Hat and SuSE support. On x86 64-bit Linux VMware Server as a native 64-bit application is run. For the Linux software as RPM package or tar.gz archive will be available. SuSE and Red Hat to install the RPM package should be done.

```
Host ~ # rpm-Uvh VMware-server-*.rpm
```

For other Linux distributions the tar.gz archive is used. Before the kernel header sources and the development tools are installed. For example, my installation on Ubuntu 8.04 LTS:

```
Host ~ $ sudo apt-get install linux-headers-`uname -r` gcc make
```

After downloading the archive is unpacked and the script *vmware-install.pl* run.

```
Host ~ $ tar xzvf VMware-server-*.tar.gz
Host $ cd vmware-server-distrib
Host ~ $ sudo ./Vmware-install.pl
```

On most issues, the default values with [Enter] to take effect.

```
...
Please specify a port for remote connections to use [902] [Enter]
Please specify a port for standard http connections to use [8222] [Enter]
Please specify a port for secure http (https) connections to use [8333] [Enter]
In which directory do you want to keep your virtual machine files?
[/ Var / lib / vmware / Virtual Machines] [Enter]
...
Please enter your 20-character serial number.
Type XXXXX-XXXXX-XXXXX-XXXXX or 'Enter' to cancel: XXXXX-...
...
```

During installation, the script *vmware-config.pl* configuration to start. This script is also after each kernel update the host system access. If the configuration with default values desired, the *option-d*.

```
Host ~ # /usr/bin/vmware-config.pl-d
```

There are kernel modules *vmmon* and *vmnet* the installed and the startup script */etc/init.d/vmware* created. Importing from CD / DVDs is at the URL http://qemu-buch.de/d/Speichermedien/Images_anlegen described. The ISO images are in the *data store* to store. Under Linux, it is by default the path is */var/lib/vmware/Virtual Machines*. If the system time at the guest systems are problems, then the host system, the boot option *acpi = off* to start with.

```
# / Boot / grub / menu.lst ... title Ubuntu 8.04.2, kernel 2.6 root (hd0, 0) kernel / vmlinuz-2.6 rc
```

The web interface VI Web Access

The web interface *VI Web Access* is used to manage virtual machines. Unless otherwise specified in the configuration, the web server listens on ports 8222 (HTTP) and 8333 (HTTPS). If the VMware Server and the Web browser on the same machine, *localhost* is used to:

- <http://localhost:8222>
- <https://localhost:8333>

If accessed from another computer, the computer name or IP address. External is only HTTPS (port 8333) allowed. The port 8333 is enabled it may be necessary in the firewall on the computer with VMware Server.

- <https://my-vmware-server:8333>

With HTTPS security check appears in a web browser, because a self-signed certificate in *VI Web Access* is used. In the Web browser of this certificate is acceptable. After logging in as *root* (Linux) or *Administrator* (Microsoft Windows) with the appropriate system password appear on the left side of the *Inventory* of the *VI Web Access* all created virtual machines. By selecting the tabs at the top on the right side displays the functions available. Under the *Console* tab gives access to the host system. This *console* is the browser plug-install *VMware remote*. The web interface tries to authenticate using a client certificate to carry out. If the Web browser Firefox, a client certificate is installed, you can switch the annoying query in the browser as follows: *Edit*, *Preferences*, *Advanced*, *encryption*, *certificates*, one select *Automatic*.

Configuration

To configure VMware Server is left in *inventory*, the host computer to select. The tab *summary* are on the right side of the *Configure Options*. *Edit Settings* with *host* machines, the allocated memory for the virtual set. Furthermore, the creation of snapshots in the background can be activated. With *Edit Virtual Machine Startup / Shutdown Settings* system is the automatic high and shut down the virtual machine at boot or shutdown the host can be activated. In order to start the virtual machines are not all at the same time, a particular delay is adjustable. Additional disk space to store the virtual machine is added with *add datastore*. If multiple users on the VMware Server working, roles can use the menu item *Administration*, *Manage Roles* are configured. The tab *permissions* are distributed to each of Rights.

Creating virtual machines

The creation of virtual machines with the web interface *VI Web Access* supports a wizard. This is accessed via the menu item *Virtual Machine*, *Create Virtual Machine*. In the first step of the virtual machine name must be indicated. In this example, the virtual machine called *WinXP* because Windows XP system is installed as a guest. This is the next step in the operating system requirements *Windows operation system* and *Microsoft Windows XP Professional* select. As the amount of RAM 512 MB is to be specified. In the next steps for the virtual disk is defined. This *disc* is *Create a New Virtual* to click and enter a size in GByte *Capacity* 8th When *On* is *bridged* network and *Connect at Power* to activate. The installation CD is provided to the virtual machine. recommend to work is, to them as ISO image and import (*Use an ISO image* and *Connect at Power On*). On a floppy drive can be omitted. USB, however, is activated. At the end you get an overview of the configuration. Click *Finish* to start creating the virtual machine. With the menu item *Virtual Machine*, *Power On* is selected virtual machine started. It switches to the *Console* tab. During the first use of the console by clicking *Install* the browser *plug-in* to install *VMware Remote Console plug*. Next, open every click in the console area and a new window with the console. This window is independent of the Web Interface. Closing the window, the *VMware Remote Console* does not end the virtual machine. The guest system will be installed as usual. To exit the window of a virtual machine, the key combination [Ctrl] + is [Alt] to enter. The virtual machine appears to the Government in the left list.

Virtual Appliances

Fully installed virtual machines (virtual appliances), drawn by the URL <http://www.vmware.com/appliances/directory/> download. You look either in the categories, for example, *operating system*, or using the search box. After downloading the archive in the Data Store is extracted. Can import the virtual machine with the menu item *Virtual Machine*, *Add Virtual Machine to Inventory*.

VMware Tools

It is recommended that you install on the guest systems, the VMware Tools. They optimize with special drivers to work with the virtual machine. It adapts to the host window to the resolution and allows *cut and paste* between host and guest. The VMware tools for the different host systems as ISO images (*freebsd.iso*, *linux.iso*, *netware.iso*, *windows.iso*) are offered. Therefore he must be a virtual machine a virtual CD / DVD-ROM drive. If this does not exist, you add it to the off virtual machine. To install the VMware Tools virtual machine to start. The tab *summary* choose Options from the *Install VMware Tools*. Here, a virtual CD-ROM is inserted into the virtual CD drive. Under the *Console* tab, you open the console, the system can serve to guests.

Microsoft Windows guest systems

On Microsoft Windows as guest system, switching to the CD drive and launch the setup program if it was not started by

startup.

Linux guest systems

On Unix / Linux as guest system is CD-ROM to mount it first, to unpack the tar archive and the script *vmware-install.pl* call. In Ubuntu 8.04 the following commands are necessary.

```
Guest ~ $ sudo mount /dev /cdrom
Guest ~ $ sudo cp / * media/cdrom0/VMwareTools.
Guest ~ $ sudo umount /dev /cdrom
Guest ~ $ sudo tar xzvf *.tar.gz VMwareTools
Guest ~ $ cd vmware-tools-distrib
Guest ~ $ sudo ./Vmware-install.pl-d
```

For Unix like host systems (VMware is to monitor the tools with Monit <http://mmonit.com/monit/>) is recommended. starts Monit stopped VMware Tools. The installation of the package *monit* Ubuntu using a command line does under.

```
Guest ~ $ sudo apt-get install monit
```

is then in */etc/default/monit startup = 0* to the line *1 = Change startup*. Monit is configured with the configuration file */etc/monit/monitrc*.

```
# / Etc / monit / monitrc
# Email Alerts for
set mail-format {
    from: root@mygast.mydomain.de
    subject: [$ ACTION] $ HOST $ SERVICE $ EVENT - monit
}
set alert my-email@mydomain.de
# # Vmware-tools
check process vmware-tools with pidfile /var/run/vmtoolsd.pid
my-email@mydomain.de alert only on nonexistent {}
start program "/etc/init.d/vmware-tools start"
stop program "/etc/init.d/vmware-tools stop"
if 5 restarts within 5 cycles then timeout
```

Then the configuration is to test and Monit to restart.

```
Guest ~ $ /etc/init.d/monit syntax
Control file syntax OK
Guest ~ $ /etc/init.d/monit
```

Control of the running virtual machine

Functions to control the running machine can be found in the web interface menu, *Virtual Machines* and under the rider right in *Summary Commands*.

Full screen

Thus the *Remote Console* window fills the entire screen, *fashion* is the menu item *Virtual Machines*, *Enter* select *Full Screen*. With a small bar at the top of the screen can be controlled by the virtual machine.

Power Off, Suspend, Suspend Guest, Guest Shutdown, Reset, and Restart Guest

On the menu item *Virtual Machines* are the commands *Power Off*, *Suspend*, *Suspend Guest*, *Guest Shutdown*, *Reset*, and *Restart Guest* available. *Power Off* and *Reset* should not be used, or threatened loss of data. Are the VMware Tools installed, you drive the virtual machine with instant-messaging *Machines*, *Shut Down Guest* cleanly shut *Virtual*. A restart with instant-messaging *Virtual Machines*, *Guest* initiate *restart*. In *Suspend* the state of the virtual machine is stored before it is finished. The next time you start this saved state is restored. In *Suspend Guest* is a script before the VMware Tools in the guest system runs separately. With *Virtual Machines*, *pause*, keep to the virtual machine, or it can continue.

access from removable media

Removable media such as CD / DVD-ROMs and diskettes, are the *Remote Console* from the menu *Devices* made available in (see the *hardware configuration of the virtual machine*).

Snapshots

Snapshots save the states of the entire virtual machine (CPU, memory, devices and recordable discs). In this way, changes can be made to the system back. A snapshot is in the running virtual machine via the menu item *Virtual*

Machines, Take Snapshot created. To restore the snapshot restore *Machines* is the menu item *Virtual*, select *Revert to Snapshot*. Since backup points need much space, *snapshot* are no longer needed protection points with *Virtual Machines, Remove* to delete.

Hardware configuration of the virtual machine

To configure the virtual machine should be shut down. New hardware is under *Summary tab commands* to add the item added to the *hardware*. The existing hardware configuration takes place under the *Summary tab* under *Hardware*. The respective icon to click, and select a point in the menu that appears.

Processor

It is the number of emulated CPUs pretend (SMP).

Memory

The amount of memory (RAM) of the virtual machine is configured here.

Hard Disk

A hard disk can be removed from virtual machine of the (*Remove*). Thus, the virtual hard drive file system of the host is deleted, select *Delete from Disk*. Edit the configuration settings using *Edit*. It can change the adapter. Can continue to operate with *independent fashion* regardless of the snapshot disk. The caching can be used for higher security (*Optimize for safety*) or requests (*Optimize for performance*) are optimized for fast.

CD / DVD

Access to CD / DVDs, the *VMware Remote Console* via (*Client Media*). If an ISO image or CD / DVD drive of the host system the host system will be made available to, is *host Media* activate. Then *Connect at power on* is to activate and physical drive or ISO file to select it.

Network Adapter

The following types of networks to choose from.

- *Bridged*: The virtual machine gets an IP address either from a DHCP server on the network or from the Internet provider. If no DHCP server is manually free IP address in the network of the host system to specify a. *Bridged* is most often used.
- *Host Only*: It is private network to the host system formed a. Outside of the host system, this network is not visible. The internal DHCP server assigns the guest system automatically assigns an IP address.
- *NAT*: *Network Address Translation* is at the internal IP address of the host system to the outside with the IP address of the host system to map the. The internal DHCP server assigns to the host system an IP address from 192.168.81.0. Also, the routing through the provided gateway is defined. A network configuration for most operating systems is not necessary and access from the host system to the outside is now available. The firewall blocks all incoming connections to this virtual network. Therefore it can not be accessed from outside the virtual machine. The host system is protected.

After installing the VMware Server VMware two new network cards are set up. On Linux, the *ifconfig* command provides information about these adapters. On Microsoft Windows Control Panel is to call the (*Start Menu, Control Panel*). Under *System, Hardware* tab you open the *Device Manager*. Under *Network Adapter VMnet8* the two VMware Network Adapter *VMnet1* and listed. The configuration of virtual networks is done with the *Virtual Network Editor*. This is accessed via the Windows *Start menu, Programs, VMware, VMware Server, Virtual Network Editor*. It can be set up networks for each virtual network adapter. These include the type of network (*Bridged, NAT, host-only*), the IP range and the internal DHCP server. Change affects all of the virtual network adapter connected virtual machines.

Other configurations of the virtual machine

To configure the virtual machine must be shut down. The link is found under the tab *Summary, Commands, Configure VM*. The following functions:

General

Here, the name of the virtual machine, operating system and the default location to be changed.

Power

The automatic full screen mode after the start is activated. The functions of some icons to control the virtual machine (*power off, suspend, ...*) can be changed. The version of VMware Tools scripts can be adjusted. To start the virtual machine into the BIOS to get the next screen is *Enter the BIOS setup* to activate. Furthermore, lassen VMware Tools update before booting and enable time synchronization with the host system.

Snapshots

The snapshot can be protected from changes (*Lock this snapshot*) and turning off the virtual machine can automatically be re-established state of the snapshot.

Advanced

This lets you make special settings.

Command Line Tools

Virtual machines can also be controlled from the command line of the host system. The command-line tools in Linux are in the directory `/usr/bin` and on Microsoft Windows in the `C:\Program Files\VMware\VMware Server`.

vmrun

With `vmrun`, virtual machines, start and stop list. Furthermore, you put it in a snapshot or suspend the virtual machine into suspend mode. The following command lists all running virtual machines (eg Microsoft Windows).

```
Host C: \> cd C:\Program Files\VMware\VMware Server
Host C: \> vmrun list
```

With the option and `start` the path to the VMX file to start a virtual machine (such as Microsoft Windows).

```
Host C: \> vmrun start c:\Virtual Machines\WinXP\WinXP.vmx
```

If no option is specified, all options are listed.

```
Host C: \> vmrun list
```

vmware-vdiskmanager

Documentation: http://www.vmware.com/support/ws55/doc/ws_disk_manager_examples.html

With VMware Server, the command line tool `vmware-vdiskmanager` included. This allows virtual disks convert create and enlarge. The `option-h` is allowed to display the options (eg Microsoft Windows).

```
Host C: \> cd C:\Program Files\VMware\VMware Server
Host C: \> vmware-vdiskmanager-h
```

In order to enlarge a virtual hard drive, the `option-x` to use. In this example, the image is enlarged to 20 GB.

```
Host C: \> vmware-vdiskmanager-x 20GB Platte.vmdk
```

Other examples can be found at the URL http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_VMware-Workstation .

Other Tools

vCenter VMware Converter

Download: <http://www.vmware.com/products/converter/>

With the free *VMware Converter* can vCenter physical computer with Microsoft Windows NT4 SP4 +, 2000, XP, Server 2003, Vista and 7 virtual machines convert. Furthermore, the following Linux distributions are supported: Red Hat Enterprise, Red Hat Advanced Server, SUSE Linux Enterprise Server and Ubuntu. The transfer of an operating system from one physical machine to a virtual machine is called physical to virtual (see http://qemu-buch.de/d/Speichermedien/_Physical-to-Virtual). For a running operating system runs with applications and data without reinstalling old hardware into a virtual machine. *VMware vCenter Converter* also allows you to move from host systems, another virtualization solution, such as Virtual PC, to a VMware product. The conversion of virtual machines is also referred to as a V2V migration (see http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware). A description of the *VMware vCenter Converters* can be found at the URL http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_VMware-Player .

vmbuilder

If the host system is Ubuntu Linux, the machine is creating a virtual alternative to the `vmbuilder` possible. The `vmbuilder` is a tool for creating virtual machines with Ubuntu JeOS as the guest system for KVM, QEMU, VMware Server and VMware Workstation, VMware Player, and Xen. Ubuntu JeOS (Just Enough Operating System) is a stripped down version of the Ubuntu Linux distribution for virtual appliances. Requires package the `python-vm-builder`.

```
Host ~ \$ sudo apt-get install python-vm-builder
```

In the following simple example of a virtual machine with Ubuntu Hardy with default values will be generated.

```
Host ~ $ sudo vmbuilder VMServer hardy
```

In the current directory the script specifies a subdirectory of the virtual disk and configuration file *ubuntu.vmx*. Can import the virtual machine in VMware Server with the menu item Virtual Machine, Add Virtual Machine to Inventory. Numerous other options *vmbuilder* allow better configuration of the new virtual machine. The option - *help* explain this.

```
Host ~ $ vmbuilder - help
```

The options of typing do not have to hand, the information in the following web form to be entered:
<http://people.ubuntu.com/~kirkland/ubuntu-vm-builder.html>. The first field is the virtualization software *VMServer* select. After entering the desired values and after pressing the button *Generate Command* command line is the corresponding display. This is copied to the console. A more detailed description can be found at the URL http://qemu-buch.de/d/Managementtools/_libvirt-Tools/_Die_Managementtools.

VMware VirtualCenter Server

The *VMware VirtualCenter Server* (<http://www.vmware.com/de/download/>) is a powerful management software for ESX and ESXi. For testing, a time-limited evaluation version can be downloaded. With the evaluation version, it is possible to manage VMware Server 2.0. The *VMware Virtual Center Server* is described in Section ESXi (see http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulatoren/_VMware_ESXi_Hypervisor]).

openQRM

openQRM is an Open Source developed since 2006 diverse suite for system management (see http://qemu-buch.de/d/Managementtools/_OpenQRM). With its plug and *vmware-server* *vmware-server2* allow virtual machines to VMware Server and VMware Server 2 to manage. To virtual machine with *openQRM* create one, plug-in is the first *VMware server2* to install and activate. Then *host* is an appliance with the resource type *VMware Server2* produce. With the *VMware Server2-manager* will be a new virtual machine.

The libvirt library

Website: <http://libvirt.org>

The library is a layer between *libvirt* virtualization software and management tools. Thus, the development of management tools for all virtualization solutions that support this library possible. The *libvirt* library supports QEMU, KVM, VirtualBox, VMware ESX, Xen, LXC Linux container system, OpenVZ Linux container system, and User Mode Linux *libvirt*. Is published as open source. For various Linux distributions, corresponding software packages are installed. The library can be compiled with Cygwin and Mingw under Microsoft Windows. More information can be found at the URL http://qemu-buch.de/d/Managementtools/_libvirt-Tools. Here is an example of a remote connection via SSH to a VMware Server computer (Linux) with the name *Pluto* and the *one I use*.

```
~ $ Virsh gsx-c + ssh: / / I @ pluto /
```

Other

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware. Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems. For server consolidation, VMware ESXi (see instead of the free VMware Server better Kernel-based Virtual Machine, the free http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulatoren/_VMware_ESXi_Hypervisor) or the VMware ESX can be used. Requires special features, such as live migration, emulation of other processors, API emulation, injections of hardware faults, debugging and manual settings of the system time is QEMU (additional) use.

<<< | # # # | >>>

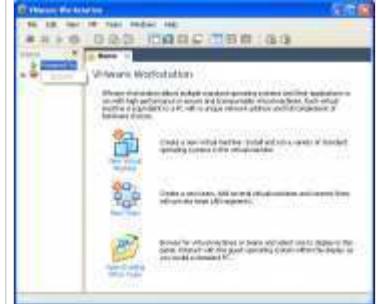
Retrieved from "http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_VMware-Server"

This page has been accessed 12,892 times. This page was last updated on 27 September 2010 at 06:56 clock changed. Content is available under GNU Free Documentation License 1.2 .

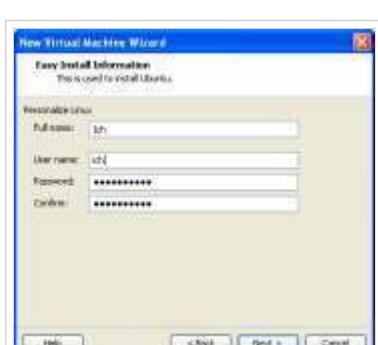
VMware Workstation 7.1 download, Serial Number, ACE import, xp-mode, vmware-vdiskmanager

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and Emulators / VMware Workstation]])

<<< | # # # | >>> | English



The VMware Workstation.

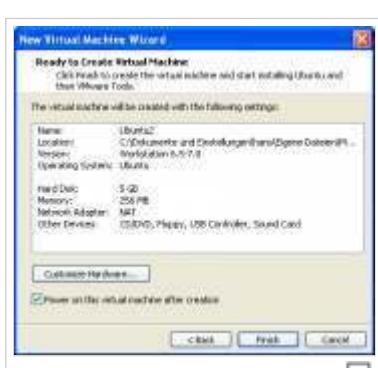
New virtual machine:
1. Step. New virtual machine:
2. Step. New virtual machine:
3. Step.



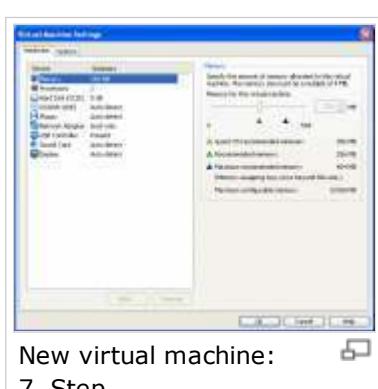
New virtual machine:
4. Step.



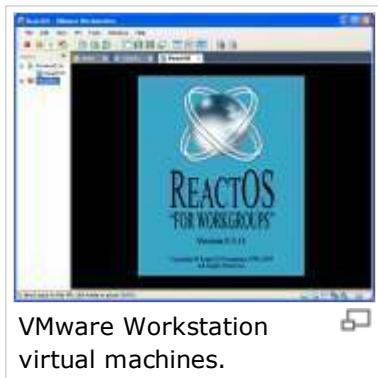
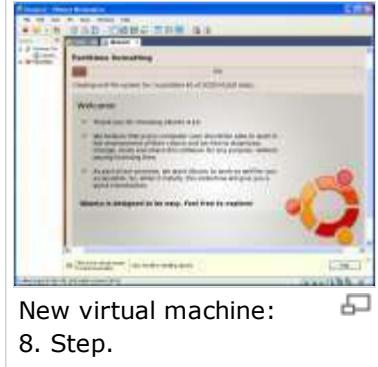
New virtual machine:
5. Step.



New virtual machine:
6. Step.

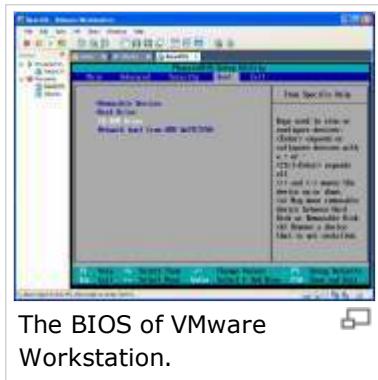


New virtual machine:
7. Step.

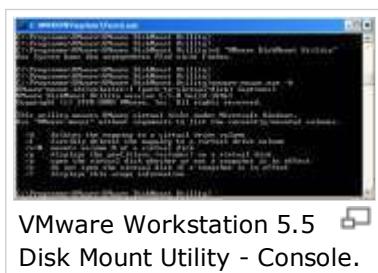




VMware Virtual Network Editor.



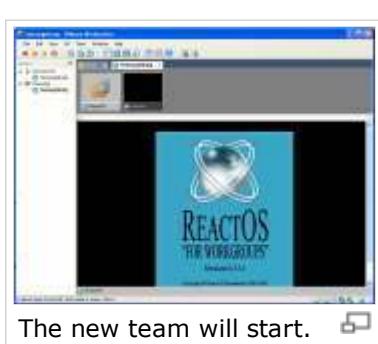
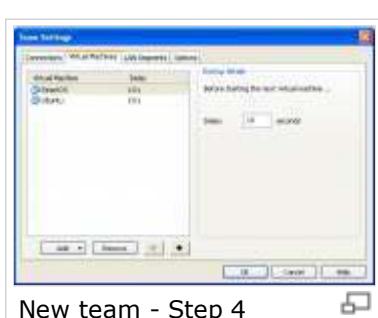
The BIOS of VMware Workstation.



VMware Workstation 5.5 Disk Mount Utility - Console.



New team - Step 1



Contents

- 1 VMware Workstation 7.1
 - 1.1 Installation
 - 1.1.1 Microsoft Windows
 - Linux 1.1.2
 - 1.2 Configuration
 - 1.3 Creating Virtual Machines
 - 4.1 Virtual Appliances
 - 1.5 virtual machines and emulators of other virtualizer
 - 1.6 XP-Mode Import
 - 1.7 Physical-to-Virtual (P2V)
 - 1.8 VMware Tools
 - 1.8.1 Windows-guest systems
 - 1.8.2 Linux guest systems
 - 1.9 control of the running virtual machine
 - 1.9.1 Full Screen
 - Unity 1.9.2
 - 1.9.3 Shutdown, Restart Guest, Suspend, Reset, Power Off and Pause
 - 1.9.4 Access from removable media
 - 1.9.5 Screenshots and Videos
 - 1.9.6 Snapshots
 - Clone 1:10
 - 1.11 Hardware configuration of the virtual machine
 - 1.11.1 Memory
 - 1.11.2 Processor
 - 1.11.3 Hard Disk
 - 1.11.4 CD / DVD
 - 1.11.5 Floppy
 - 1.11.6 Network Adapter
 - 1.11.7 USB Controller
 - 1.11.8 Sound Card
 - 1.11.9 Display
 - 1:12 Additional configurations of the virtual machine
 - 1.12.1 General
 - 1.12.2 Power
 - 1.12.3 Shared Folder
 - 1.12.4 Snapshots
 - 1.12.5 AutoProtect
 - 1.12.6 Replay
 - Guest Isolation 1.12.7
 - 1.12.8 Encryption
 - 1.12.9 Tools
 - 1:12:10 Remote Display
 - 1:12:11 Unity
 - 1:12:12 Appliance View
 - 1:12:13 ACE
 - Advanced 1:12:14
 - 1.13 The virtual BIOS
 - 1:14 Teams
 - 1:15 VMware Workstation 5.5 Disk Mount Utility
 - 1:16 vmware-vdiskmanager
 - 1:17 Other



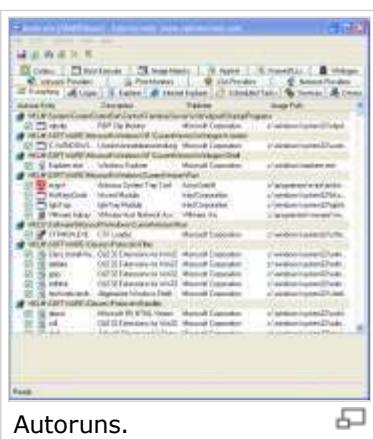
VMware Workstation 5.5
Disk Mount Utility -
Step 1



VMware Workstation 5.5
Disk Mount Utility -
Step 2



CCleaner cleans
Microsoft Windows.



Autoruns.

VMware Workstation 1.7

Download: <http://www.vmware.com/products/workstation/index.html> (30 day demo version after

registration)

The paid VMware Workstation (from VMware Inc.) is based on the native virtualization, and supports full virtualization. It enables on computers with x86 or x86-64 processors, multiple virtual machines for these processors to use. The installation of supported guest systems is comfortably possible with *Easy Install*. On Microsoft Windows 7, the VMware Workstation import the XP mode. VMware Workstation supports the Open Virtualization Format (OVF) and the Open Virtual Appliance (OVA). The VMware Workstation is *Microsoft Virtual PC* and *Microsoft Virtual Server* and Virtual Open operate machinery. VMware Workstation continues to support the image of *Symantec Backup Exec System Recovery* (formerly *LiveState*).

Virtual machines can be a copy (*clone*) are generated. VMware Workstation supports the creation of snapshots and a team function. The Teams feature allows you to group virtual machines and connect to virtual networks. The function *Unity* applications run seamlessly in the host system directly to the desktop of the host system. *Copy & Paste* is on between host and guest system possible. *Drag and drop* files between host and guest system is supported. VMware Workstation supports the Aero desktop, Microsoft Windows 7 In the guest system to print directly to the printer connected to the host system. VMware Workstation supports the creation of packages with virtual machines for VMware fee-ACE (Assured Computing Environment). These virtual machines and the VMware Player Distributed be provided with security policies and ACE Management Server can be managed centrally by one. Furthermore, portable virtual machines can be created for USB storage media. VMware Workstation runs on Linux and Microsoft Windows. On 32-bit versions of Windows can not run the VMware Workstation parallel with VMware Server.

Installation

To install a paid license key is available for purchase. It is possible to test this software for 30 days. This URL is a license key for the evaluation version at the <http://www.vmware.com/downloads/ws/eval.html> request. Entering the serial number is on the menu *Help, Enter Serial Number* possible.

Microsoft Windows

After downloading the file to start *vmware-workstation-full-*.exe*. In step *setup type* do you choose *Typical*. Then C is the target *directory*: \ Program Files \ VMware Workstation \ preset \. With *Change* this setting can be changed. Then asked if *shortcuts* are created to call for the comfortable. These settings do not change. At the end of the installation the serial number is entered. Finally, Microsoft Windows is restarted. Is called the VMware Workstation from the *Start menu, Programs, VMware, VMware Workstation*.

To import the Microsoft Windows CD and DVD as an image, additional software is required. Most CD / DVD burning programs support importing from CD / DVDs. A free program is *ImgBurn* (<http://www.imgburn.com>). It loads the installer *SetupImgBurn_*.exe* from the site now and start it. After starting the program, choose *Create image file from disc* to ISO image of the install disc to create a.

Linux

To install the kernel header sources and the development tools necessary (eg Ubuntu 9.04) are.

```
Host ~ $ sudo apt-get install linux-headers-`uname -r` gcc make build-essential
```

It is the latest version of VMware Workstation (32 - or 64-bit) download. The graphical installer is started as follows.

```
Host ~ $ chmod +x ./Vmware-Workstation-Full-7.*.bundle
Host ~ $ gksudo bash ./Vmware-Workstation-Full-7.*.bundle
```

The installation is self explanatory. It installs special kernel modules. Then is called VMware Workstation.

```
Host ~ $ /usr/bin/vmware &
```

Importing from CD / DVDs is at the URL http://qemu-buch.de/d/Speichermedien/_Images_anlegen described.

Configuration

Via the menu item *Edit, Preferences* VMware Workstation is configured. The tab *workspace* directory is specified, the new virtual machines are stored in the. Under the *Hot Keys* tab key combination to adjust, with the one mouse from the virtual machine window, free the (Default: [Ctrl] + [Alt]).

Creating virtual machines

In the VMware Workstation *New Virtual Machine* is the icon, a wizard to create a new virtual machine is started with. The first step is *Typical* and then select the installation media. With *Installer disc*, a DVD in the host system inserted CD / used. In this example, the installation of Ubuntu to an ISO image. The ISO file to *installer disc image file* to select. For supported host systems using the VMware Workstation *Install* a technique called *Easy*. This installation is largely automatic. *Easy Install* on Ubuntu asks at the beginning only after the user to be created. The next step in the name of the virtual machine and the target directory is specified. Subsequently, the size of the virtual disk is to be specified. Larger boards are better to invest in several parts. In the overview at the end of the wizard makes the *hardware Customize* with virtual hardware to adapt. With the *Finish* button to start the installation of the host system in the virtual machine. The demand for the installation of VMware Tools must be answered positively. The new virtual machine is next listed under *Favorites*. Start the virtual machine via the menu item *VM, Power, Power On*. In order to have control of the host system, you have to click in the window of the virtual machine. To return to the desktop of the host system, the key combination [Ctrl] + is [Alt] necessary.

If a virtual machine is often used to set up a quick start shortcut. To do this click the right mouse button on the Windows desktop and choose *New, Shortcut*. *Browse* to choose VMX file of the desired virtual machine from the. These are usually in *My Documents \ My Virtual Machines*. Subsequently, this linkage should be designated. With a double click on the link you start the virtual machine.

Virtual Appliances

Fully installed virtual machines (virtual appliances), drawn by the URL <http://www.vmware.com/appliances/> download. You look either in the categories, for example, *operating system*, or using the search box. After downloading the archive is unpacked. Can import the required virtual machine either via *File Open* or *File, Import or export*. With *File, Open* the import is faster because default values are taken. In *File, Import or Export Wizard* will be launched. In the wizard choose the point *source type of virtual appliance* on.

Virtual machines and emulators of other virtualizer

Virtual machines and emulators of other virtualizer either *File, Open* or *File, Import or export* of imported. With *File, Open* the import is faster because default values are taken. In *File, Import or Export Wizard* will be launched. In the Wizard, choose *Other* from the point *Source Type* in. The next step is to open the file. Will virtual machines from *Microsoft Virtual PC* or *Microsoft Virtual Server*, open the VMware Workstation generates a new VMX file. The original VMC file remains unchanged. Images are from *Symantec Backup Exec System Recovery* open, generates a new VMware Workstation VMX file. The original image is unchanged. Except for the formats. Ovf and. Ova *Linked clones* are created here. In the formats. Ovf and. Ova are *full clones* generated. Microsoft Windows as guest system detects at boot time after importing new hardware and install default drivers for it. Then a restart of the host system is necessary.

With *qemu-img* (see http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware) and *ubuntu-vm-builder* (see http://qemu-buch.de/d/Managementtools/_libvirt-Tools/_Die_Managementtools) may also converts virtual machines are created.

XP-Mode Import

The XP-mode Microsoft Windows 7 Professional, Ultimate, and Enterprise is a free image with a custom Windows XP Professional. This virtual machine is normally powered by Microsoft Virtual PC 2007. After the installation of VMware Workstation is the URL of the XP-Mode <http://www.microsoft.com/windows/virtual-pc/download.aspx> download. Then file is the XP mode by calling the *us.exe* to install *WindowsXPMode_de*. After installing *XP*, the XP-Mode in VMware Workstation with *File, Import Windows VM* imported fashion. The VMware Workstation finds the XP mode and rename the new virtual machine with *Windows XP mode*. After starting the virtual machine, the VMware Tools are installed.

Physical-to-Virtual (P2V)

The process of transferring real computers into virtual machines is called physical to virtual (P2V). An appropriate Wizard, the *File menu, import or export* started from. It is on the point *computer* select *Source Type Physical*. The next step is to specify whether it is the local or another computer. Is it a remote computer, the name or IP address and the password is specified. Then to copy media must be selected. The next step, the target system is to be specified. Finally, the name and location is to define the virtual machine.

Alternatively, use the VMware Converter for P2V vCenter migration. A description can be found at the URL http://qemu-buch.de/d/Anhang/_Weitere_Virtualisierer_und_Emulator/_VMware-Player.

VMware Tools

If not done, the VMware Tools are installed. This is a CD image with various drivers. This can be set as higher screen resolutions. A common buffer is used. Another advantage is the easy exchange of the mouse pointer between the desktop and the virtual machine window. The installation is the running virtual machine via the menu option *VM, install VMware Tools* introduced in. Here, the virtual CD to the virtual CD drive on the host system is loaded. The CD image with VMware Tools is loaded on first use of the VMware Web site. More information can be found at the URL <http://www.vmware.com/support/pubs/>.

Windows guest systems

Microsoft Windows is usually the case when you insert the virtual CD automatically called in the file *setup.exe*. If it fails, the automatic start, so manually in the file manager the file *setup.exe* to start. It is typical to select the installation type. Then, the host system is restarted. To set a higher screen resolution in the guest system, just click the right mouse button on any free space on the Windows desktop. Under the *Settings* tab to configure the screen resolution.

Linux guest systems

On Unix / Linux as host system, this CD-ROM is included. This is normally done automatically. The tarball is unpacked and the script *vmware-install.pl* call. They are the kernel headers and development tools necessary. On Ubuntu, the following commands are entered. After each kernel update, the script re-run *vmware-install.pl*.

```
Guest ~ $ sudo cp / * media/cdrom1/VMwareTools.
Guest ~ $ sudo umount / dev / cdrom
Guest ~ $ tar xzvf *. tar.gz VMwareTools
Guest ~ $ cd vmware-tools-distrib
Guest ~ $ sudo . / Vmware-install.pl-d
```

Control of the running virtual machine

Functions to control the running machine can be found in the *VM* menu. For example, one sends the host system, the key combination [Ctrl] + [Alt] + [End] with instant-messaging *VM, Send Ctrl + Alt + Del*.

Full screen

This virtual machine window fills the entire screen, the menu item *View, Full Screen* or press [Ctrl] + [Alt] + [Enter] to open. With a small bar at the top of the screen can be controlled by the virtual machine. With the menu item *View, Quick Switch* or [F11] under the *Home* tab on the virtual machine fills the entire screen, too. However, at the top of the riders remain for switching between virtual machines.

Unity

The Unity feature lets you merge the desktop of the host system to the desktop of the host system. The applications of the host system running as a window on the host system. The Unity feature is available after installing the VMware tools. In the running virtual machine *Unity* is the *View menu*, activated by *Unity*. The start menu of the host system is activated via the key combination [Ctrl] + [Shift] + [V].

Shutdown, Restart Guest, Suspend, Reset, Power Off and Pause

With the menu-point *VM, Power*, the commands are *reset*, *suspend* and *power off* available. *Reset* and

Power off should not be used, or threatened loss of data. Are the VMware Tools installed, you drive the virtual machine with instant-messaging *VM* down cleanly *shut down Guest*. A restart is instant-messaging *VM, Restart Guest* him to arrange. In *Suspend* the state of the virtual machine is stored before it is finished. The next time you start this saved state is restored. With *VM, pause*, keep to the virtual machine, or it can continue.

access from removable media

Removable media such as CD / DVD-ROMs, floppy disks and USB flash drives, are the virtual machine menu option *VM, Removable Devices* made available via.

Screenshots and Videos

With instant-messaging *VM, Capture Screen* system is a screen shot of the guest in the clipboard. Via the menu item *Edit, Preferences* tab is the *workspace* under a different goal set. With instant-messaging *VM Capture Movie* recording is a video starts. This is AVI file *under Registered Documents, My Videos* stored as. To finish the recording with *VM, Stop Movie Capture*.

Snapshots

Snapshots save the states of the entire virtual machine (CPU, memory, devices and recordable discs). In this way, changes can be made to the system back. A snapshot of the running virtual machine via the menu option *VM, Snapshot, Take snapshot* created in. Here, the snapshot is named. Optional description, a *description* entered under. Snapshots are managed in the *Snapshot Manager*. This is accessed via *VM, Snapshot, Snapshot manager*. To restore a snapshot, restore, select one of these and click on *Go To*. Since backup points need much space, protection points are no longer required to delete with *Delete*. About *Auto Protect* can create snapshots at specific time intervals automatically.

Clone

With the menu-point *VM, clone* a virtual machine is powered down copies a. Here, the current state or condition of a snapshot can be used. Alternatively, clone the *Snapshot Manager* on the *Clone* button generates a. Copying can be done in two ways. For a full clone (*Create a full clone*) copy is a true, independent created. When a linked clone (*Create a linked clone*) is the original virtual machine or its snapshot, the basis for the new machine. This is the overlay files of QEMU (see comparable http://qemu-buch.de/d/Speichermedien/_Images_anlegen). A linked clone can be operated together with the original virtual machine. The linked clone is created quickly, but offers a poorer performance.

Hardware configuration of the virtual machine

To configure the virtual machine should be shut down. The corresponding dialog box you call the menu *VM, Settings*. Under the *Hardware* tab, the virtual hardware is configured. New hardware is added using the *Add* button. It helps a wizard. Supported Linux and Windows guest systems can print directly to the printer on the host system. *Add* to this is to use *Virtual Printer*. The guest system configures the printer automatically.

Memory

The amount of memory (RAM) of the virtual machine is configured here.

Processor

It is the number of emulated CPUs pretend (SMP). The settings under *Virtualization engine* usually do not need to be changed. For special guest systems and test environments, the type of virtualization / emulation is adapted.

Preferred mode:

- *Automatic*: It tries to automatically host and guest system to find the optimal code for that. This includes *Automatic with replay*, *Binary translation*, *Intel VT-x or AMD-V* and *Interl-VT-x/EPT* or a *AMD-V/RVI*. The other options is the *preferred mode* explicitly defined.
- *Automatic with Replay*: Successfully optimized code is executed when needed again. This option is used when reference is made to a message.
- *Binary translation*: is a mixture of directly executing code and binary translation. This option is

using *shadow page tables* to map the memory of the host system.

- *Intel VT-x or AMD-V*: This option is only available if the host system that supports full virtualization. In addition to the appropriate hardware extensions for Full Virtualization *Shadow Page Tables* used this option to map the memory of the host system.
- *Interl-VT-x/EPT or AMD-V/RVI*: This option is only available if the host system that supports full virtualization. In addition to the appropriate hardware extensions for Full Virtualization *Paging* uses this option to map the *hardware* memory of the host system.
- *Disable acceleration for binary translation*: Some host systems do not start when this option is enabled.
- *VMware kernel paravirtualization*: <http://www.vmware.com/technical-resources/interfaces/paravirtualization.html>

Hard Disk

Among *utilities*, the following functions:

- *Map ...*: Adds the virtual hard drive as a drive in the host system. The host system must adjust the file system that drive can read. It is recommended to read to use the virtual disk (*Open file in read-only mode*).
- *Defragment*: Virtual Hard Drive Microsoft Windows guest systems can be defragmented. On Linux this is not necessary.
- *... Expand*: Enlarges the virtual hard disk. In the host system on the additional space, additional partitions are created and formatted.
- *Compact*: Optimizes a virtual hard disk.

Under *Advanced*, the behavior of the virtual disk pretend.

- *Independent*: The hard disk is independent of the snapshots.
- *Persistent*: Changes are immediately and permanently written to disk.
- *Nonpersistent*: The changes will be exiting the virtual machine or to restore lost after a snapshot.

CD / DVD

An ISO image or CD / DVD drive of the host-guest system can be made available to the system. *Connect at power on* is activated.

Floppy

An image or the floppy drive of the host-guest system can be made available to the system. *Connect at power on* is activated.

Network Adapter

The following types of networks to choose from.

- *Bridged*: The virtual machine gets an IP address either from a DHCP server on the network or from the Internet provider. If no DHCP server is manually free IP address in the network of the host system to specify a. *Bridged* is most often used.
- *NAT*: *Network Address Translation* is at the internal IP address of the host system to the outside with the IP address of the host system to map the. The internal DHCP server assigns to the host system an IP address from 192.168.81.0. Also, the routing through the provided gateway is defined. A network configuration for most operating systems is not necessary and access from the host system to the outside is now available. The firewall blocks all incoming connections to this virtual network. Therefore it can not be accessed from outside the virtual machine. The host system is protected.
- *Host Only*: It is private network to the host system formed a. Outside of the host system, this network is not visible. The internal DHCP server assigns the guest system automatically assigns an IP address.
- *Custom*: allocation of a virtual network.
- *Team*: assign a VLAN of the team.

After installing the VMware Workstation VMware two new network cards are set up. On Linux, the *ifconfig*

command provides information about these adapters. On Microsoft Windows Control Panel is to call the (*Start Menu, Control Panel*). Under *System, Hardware* tab you open the *Device Manager*. Under *Network Adapter VMnet8* the two VMware Network Adapter *VMnet1* and listed.

The configuration of virtual networks is done with the *Virtual Network Editor*. This is accessed via the menu *Edit, Virtual Network Editor* or from the Windows *Start menu, Programs, VMware, Virtual Network Editor*. It can be set up networks for each virtual network adapter. These include the type of network (*Bridged, NAT, host-only*), the IP range and the internal DHCP server. Change affects all of the virtual network adapter connected virtual machines.

USB Controller

- *Enable high-speed support for USB 2.0 devices*: This option should be enabled.
- *Automatically connect new USB devices*: new USB devices connected to the host system are automatically uploaded to the guest.
- *Show all USB input devices*: There are all USB devices to the host system of the host system shown.

Sound Card

The sound output can be disabled or set a different sound card.

Display

It can enable the 3D acceleration (*Accelerate 3D graphics*). Furthermore, the number of monitors and their resolution will be set.

Other configurations of the virtual machine

To configure the virtual machine must be shut down. The corresponding dialog box you call the menu *VM, Settings*. Under the *Options* tab, the following functions:

General

Here, the name of the virtual machine, operating system and the default location to be changed.

Power

The automatic full screen mode after the start is activated. Furthermore, the host system the battery status is transmitted. The functions of some icons to control the virtual machine (*power off, suspend, ...*) can be changed.

Shared Folder

To exchange data between the guest and host system are *shared folder* (shared folders) uses. There must have installed the VMware Tools. To set up a *Shared Folder* is *Always* select *enabled*. With the *Add* button system a path to the host selected. The shared folder is read-only system (*read-only*) assigned to the guest. In Microsoft Windows Explorer as a guest system is started and *Shared Folders* to open the entry.

Snapshots

Turning off the virtual machine can be connected with snapshot operations.

AutoProtect

About *Auto Protect* can be to create snapshots at specific time intervals automatically.

Replay

To debug this, special features such as automatic snapshots are configured.

Guest Isolation

In order for the guest system is better insulated, can *drag and drop* and *copy & paste* between host and guest system are prevented. Furthermore, the *Virtual Machine Communication Interface* (VMCI) are disabled.

Encryption

The virtual machine can be encrypted with a password.

Tools

The upgrade of VMware tools is defined.

Remote Display

Virtual Network Computing (VNC) allows the screen of a computer (on which the VNC server running) to another computer (on which the VNC viewer is running) show in exchange for keyboard and mouse movements to send back. So that the host system can access with VNC, *Enable remote display* is activated. Access is zuschützen with a password.

Unity

The behavior when *Unity* is adapted here. In addition to the configuration of the window frame, the menu of the applications (de-) activated.

Appliance View

These settings will be displayed instead of the expenditure of the host system, a description and link. This is advantageous when services are of the host system via a web server, configured as Webmin.

ACE

Features To use ACE *ACE* is enabled *Enable features* and needs an ACE server, giving at.

Advanced

Here are special settings, such as process priorities, possible.

The virtual BIOS

Each virtual machine under VMware Workstation has like a real PC BIOS (Basic Input Output System). The BIOS enables communication between the operating system and hardware. When booting the virtual machine can be reached by pressing the [F2] in the virtual BIOS. If you want to use often live CD / DVD, one is under the *boot CD-ROM drive* first. This is using the cursor keys to select *CD-ROM drive* and press the] several times to press + [.

Teams

A group of virtual machines can be defined *team* as a VMware Workstation under. This team can be a virtual network (VLAN) assigned. In each case, a network bandwidth can be defined. Virtual machines can run a team with one click in a certain order. Here, the appropriate starting place with a predetermined delay. A new team is the menu item *File, New, team* up on. In the Wizard, the name and location of the new team should be given. The next step is to add two aircraft this team to assign at least. Next, the LAN segments should be selected. Then, the virtual machine to the LAN segments to be connected. *Finish* with education, the team concluded. The new team will appear next to *Favorites*. The team settings are the menu-point *team, settings* changed over. The tab *Virtual Machines* will delay the boot sequence and configure your boot. The start is the menu item *Team, Power, Power On*.

VMware Workstation 5.5 Disk Mount Utility

For VMware Workstation *VMware Workstation 5.5* is the program *Disk Mount Utility*. This allows virtual disks on the host system (Microsoft Windows) embed. Given the virtual machine must be off. Needs to continue to the host system's file system can read the virtual machine. Via the menu item *File, Map or Disconnect Virtual Disks* program, the *Workstation 5.5 Disk Mount utility* called *VMware*. To include a virtual disk, click on *Map*. *File name* behind it is the path to the virtual disk (. Vmdk) a. In *volume* is to click the desired partition. Desktop *Drive* drive letter is free to choose. recommend) is a read-only access (*Open file in read-only mode*). In Windows Explorer to access the virtual hard disk on the specified drive letter is possible. To release we open again the program *VMware Workstation 5.5 Disk Mount Utility*. There, select the drive letter and click on the *Disconnect*. This may be no access to this drive. The program *Workstation 5.5 Disk Mount Utility* *VMware* is actually the command line tool *vmware-mount*.

vmware-vdiskmanager

Documentation: http://www.vmware.com/support/ws55/doc/ws_disk_manager_examples.html

With VMware Workstation *vdiskmanager* is also the command line tool *vmware-supplied*. This allows virtual disks convert create and enlarge. The *option-h* is allowed to display the options (eg Microsoft Windows).

```
Host C: \> cd C: \ Program Files \ VMware \ VMware Workstation \
Host C: \> vmware-vdiskmanager-h
```

In order to enlarge a virtual hard drive, the *option-x* to use. In this example, the image is enlarged to 20 GB.

```
Host C: \> vmware-vdiskmanager-x 20GB Platte.vmdk
```

When you convert virtual disks for other programs, such as QEMU / KVM, consider the following. Large virtual hard disks on VMware products, often multiple image files (.Vmdk), each with a maximum size of two gigabytes divided on. In addition, there is another file (.Vmdk) is created which, however, contains meta-data. The best way to understand the structure of a virtual disk for the example of a VMware virtual machine with an 8-gigabyte hard drive.

Case 1: The entire hard disk to an image file is located, for example *win.vmdk*. It contains all the necessary data and can easily convert to other virtualization solutions. However, was a snapshot in VMware made, this file contains only the state of the file system until the time of the snapshot. All changes made to another file (*win-000001.vmdk*) stored in. Possibly other snapshots are counted (*000002.vmdk* *win*, *win-000003.vmdk* ...). Before converting the data from the base file and snapshots are back together.

Case 2: The image file is divided into slices. In this example, next to the file in addition to files *win.vmdk* *win-win-s005.vmdk* *s001.vmdk* to present hard drive. *Win.vmdk* contains meta-data to build the virtual:

```
# Disk DescriptorFile
version = 1
CID = 793c1936
parentCID = ffffffff
create type = "twoGbMaxExtentSparse"
# Extent description
RW 4192256 SPARSE "win-s001.vmdk"
RW 4192256 SPARSE "win-s002.vmdk"
RW 4192256 SPARSE "win-s003.vmdk"
RW 4192256 SPARSE "win-s004.vmdk"
RW 8192 SPARSE "win-s005.vmdk"
# The Disk Data Base
# DDB
ddb.virtualHWVersion = "4"
ddb.geometry.cylinders = "1024"
ddb.geometry.heads = "255"
ddb.geometry.sectors = "63"
ddb.adapterType = "BusLogic"
```

Also in this case, snapshots are possible. Then the files are also *000001.vmdk* *win* and *win-000001-s001.vmdk* to *win-000001-s005.vmdk* before. Again, in *win-only* meta-data contains *000001.vmdk*.

```
Host C: \> vmware-vdiskmanager-r-t 0 000001.vmdk win-win-export.vmdk
```

This command converts the file *win-win* in the export file *000001.vmdk* *export.vmdk*. In this case, a complete file, starting from the first snapshot of type *sparse* (-t 0) is generated. This export file can be used by QEMU or KVM.

Those who work with the many options of the command line tool *vmware-vdiskmanager* is cumbersome to install itself to the VMX Builder (<http://www.vmxbuilder.com>). This package contains the GUI *Virtual DiskFaktory*. *Devfarm* started the program on the *Start menu*, *All Programs*, *software*, *VMXBuilder*, *Virtual*

DiskFaktory. In order to enlarge a virtual hard disk, for example, is chosen under the tab *Expand the virtual disk* path to the virtual disk from behind. After that is placed under the *disk size* desired size. Clicking on *Expand disc* increases the disk. be partitioned in the host system, the new area on the virtual hard disk and formatted. To drive to shrink a virtual, is under the tab the path behind *Shrink Virtual Disk* and *Disk* to specify to click *Shrink*.

Other

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware . Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems. Requires special features, such as live migration, emulation of other processors, API emulation, injections of hardware faults, debugging and manual settings of the system time is QEMU / KVM (additional) use.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_VMware-Workstation "

This page has been accessed 16,635 times. This page was last updated on 27 September 2010 at 06:57 clock changed. Content is available under GNU Free Documentation License 1.2 .

Win4BSD FreeBSD 8.0, Win4Lin Pro, Win4Solaris, download license installation howto

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / WIN4]])

<<< | # # # | >>> | English

Contents	
1 Win4BSD	
1.1 Install PC-BSD under Win4BSD	
2.1 Creating Virtual Machines	
1.2.1 With loadwinproCD and installwinpro	
1.2.2 Direct from installation media	
1.3 Management of virtual machines	
1.3.1 Usage	
1.3.2 Network	
1.3.3 Configuration	
1.3.4 International	
2 Win4Lin	
3 Win4Solaris	
3.1 Other	

Win4BSD

Download: <http://win4bsd.com/wp/get-win4bsd/>

Recommended: 512 MB RAM, 10 GB of free hard disk space, X-Window resolution of 1024x768 pixels.

Win4BSD is a commercial product from Virtual Bridges, Inc. based on QEMU. This means that the Microsoft Windows versions 2000 and XP or run individual Windows applications on FreeBSD (32-bit) and PC-BSD (32-bit). On common directories do you transfer data to the host system. Start with the configuration option is not possible. The non-commercial use of Win4BSD is free. Since this version is only a 32-bit application is for 64-bit hosts the original version of QEMU (64-bit) recommended. Other products of Virtual Bridges, Win4Lin Inc. and Win4Solaris (see below).

Installing Win4BSD under PC-BSD

In this example Win4BSD 1.1 is installed via FreeBSD ports under PC-BSD 7.0.1 (x86, 32-bit). Previously been installed on the system QEMU, and KQEMU AQEMU. Under PC-BSD and the installation of the PBI file is possible. To install via FreeBSD ports you go into the ports tree and then start the installation, which stops at PC-BSD 7.0.1 with an error message.

```
Host ~ # cd /usr/ports/emulators/win4bsd
Host ~ # make install clean
win4bsd-1.1_3 requires the traditional scheduler 4BSD. Please
rebuild your kernel with options sched_4bsd, and run make again.
*** Error code 1
```

Win4BSD 1.1 is not compatible with the new ULE scheduler. A kernel is compiled with the old BSD scheduler. This requires the kernel sources. If the directory `/usr/src/sys` does not exist, the sources must be installed. This is done using `sysinstall`. Are the sources available, you go into the directory `/usr/src/sys/i386/conf` and submit a copy of the configuration file.

```
Host ~ # cd / # cp GENERIC usr/src/sys/i386/conf host ~ MYKERNEL
```

The file is `MYKERNEL` with a text editor, eg vi, open the. The following line should be adjusted.

`SCHED_ULE` options

This line is amended as follows:

```
options SCHED_4BSD
```

After saving the file `MYKERNEL` kernel is the custom compiled.

```
Host ~ # cd /usr/src
Host ~ # make build kernel KERNCONF = MYKERNEL
```

The compilation takes a while. In the meantime, you read the documentation of Win4BSD: <ftp://ftp.vbridges.com/pub/archive/pro/releases/bsd/pro/1.1/Users-Guide.html>.

If the new kernel compiled without errors, it is installed. Then, the computer is restarted.

```
Host ~ # make install kernel KERNCONF = MYKERNEL
Host ~ # reboot
```

Win4BSD now being installed. This will return you to the appropriate directory in the ports tree and executes the installation.

```
Host ~ # cd /usr/ports/emulators/win4bsd /
Host ~ # make install clean
```

After the installation script is the `postinstall.sh` configuration and starting to run Win4BSD. This script tries to compile KQEMU, which will fail. Since KQEMU has already been installed, these error messages should be ignored.

```
Host ~ # /usr/local/lib/win4bsd/bin/postinstall.sh
Starting Win4BSD Pro ... done.
```

The launch of Win4BSD is checked with `ps`.

```
Host ~ # ps aux | grep "win4bs [d]"
root 13523 0.0 0.2 1332 832? Is 0:00,00 8:21 am
/Usr/local/lib/win4bsd/bin/mergeprod
```

The serial number `1w6fb61x-n3392a-869x-8j25-h425-31` from the download URL is entered the script `ask_license.sh` with. The data are the file `/usr/local/win4bsd/install/license.lic` stored in.

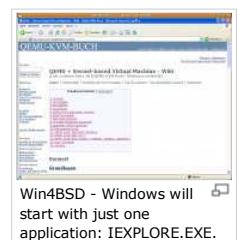
```
Host ~ # /usr/local/lib/win4bsd/bin/ask_license.sh
```

Creating virtual machines

With loadwinproCD and installwinpro

The script `loadwinproCD` certain installation media Win4WSD be imported. The following Microsoft Windows CD-ROMs are officially supported.

- Microsoft Windows 2000 Professional (no service packs)
- Microsoft Windows 2000 Professional with Service Pack 1
- Microsoft Windows 2000 Professional with Service Pack 2
- Microsoft Windows 2000 Professional with Service Pack 4



- Microsoft Windows XP Professional with Service Pack 1 or 1a
- Microsoft Windows XP Professional with Service Pack 2
- Microsoft Windows XP Home with Service Pack 1 or 1a
- Microsoft Windows XP Home with Service Pack 2

This is the CD insert and the script to start *loadwinproCD*. Under PC-BSD 7 breaks this script with an error message.

```
Host ~ # /usr/local/lib/win4bsd/bin/loadwinproCD
Checking CDROM ...
mount: / dev/acd0: Operation not supported by device
```

The script calls the script on *loadwinproCD unmount.sh* that the *mount* command is an option for using this version of PC-BSD is not clear. To handle this problem with a change in the script *unmount.sh*. You open the script */usr/local/lib/win4bsd/bin/unmount.sh* with an editor and then studied the function *mount_block_bsd* by (). In this function to change the mount command as follows:

```
mount_block_bsd ()
{
    RETVAL = 0
    if [-x / sbin / mount_auto-o-x / usr / sbin / mount_auto] then
        MTYPE = "auto"
    else
        MTYPE = "cd9660"
    fi
    # Original:
    # Mount-r-t $ MTYPE $ 1 $ 2> / dev / null
    # Re:
    mount-r-t cd9660 $ 1 $ 2> / dev / null
    if $ [? -Ne 0] then
        RETVAL = 1
    fi
}
```

It is the installation CD, in this example, Windows XP Professional (Service Pack 2) imported for Win4BSD. The image is in the directory */usr/local/win4bsd/cdroms*.

```
Host ~ # /usr/local/lib/win4bsd/bin/loadwinproCD
Checking CDROM ...
Media found: Windows XP Professional (Service Pack 2)
Loading CDROM ... please wait ...
Windows CD-ROM complete load.
```

It logs in under the normal account (not as *root* user) and the script calls on *installwinpro*. The *option-d* engine, a directory for the files of the virtual given. If this option is not specified, *WinPro* is the default directory *~/directory*. With the *-g* option specifies to set the size of the image. In the *option-m* is the mode of installation defined. There are the following:

- *image*: The entire installation is in the image *GUEST.IMG*. The BSD home directory can be reached through the path *\HOST\HOME*.
- *shared*: Unlike *image* is *My Documents* stored in the host system. This is the default setting in the Windows XP installation.
- *integrated*: as a *roaming profile* system, all user data stored in the host. This is the default setting on the Windows 2000 installation. In XP is possible under this option no access to the login screen.

```
Host ~ $ /usr/local/lib/win4bsd/bin/installwinpro \
MeinWin-d-g-m shared 10G
```

When you install Microsoft Windows only two actions are necessary: the confirmation of the license and enter the serial number. After installing a backup of the directory is recommended.

```
Host ~ $ tar czvf meinWin.tar.gz meinWin
```

Direct from the installation media

For installation media that are supported by *loadwinproCD* not, for example, Microsoft Windows XP Professional with SP3 (multi-volume CD) script is used with the following options *installwinpro*. The script *loadwinproCD* is not needed.

```
Host ~ $ /usr/local/lib/win4bsd/bin/installwinpro \
Winxppro-w-c /dev/acd0-g-m shared 10G
```

The *option-c* is one path to the CD / DVD-ROM device to the. With the *-g* option specifies to set the size of the image. In the *option-m* is the mode of installation is defined (see above). The *option-w* is the Microsoft version of Windows before:

- *win2kpro* (Microsoft Windows 2000 Professional)
- *winxppro* (Microsoft Windows XP Professional)
- *winxphome* (Microsoft Windows XP Home Edition)

This installation is also supported by GUI. Here, the installation of only one host system is possible. Whose files directory *~/WinPro* created in.

```
Host ~ $ /usr/local/lib/win4bsd/bin/mergepro-oneclick
```

Management of virtual machines

Usage

Win4BSD *WinPro* is called with. The *option-snapshot* protects the image of change. If the host system in the mode or *integrated shared* installed, files can be stored permanently anyway. This requires that the files in *My Documents* are stored. *WinPro* uses the files in the default directory (*WinPro*) when no directory is specified. In this example, the list of files used *meinWin*.

```
Host ~ $ winPRO snapshot meinWin
```

Using the [Shift] + [F12], a pop-up menu for special options to control the current instance is opened. It should be only in an emergency session to select *Exit* or *Abort* apply *session*. To stop an instance, the menu is the guest system in the start-point select the *shutdown*. If the guest system is stopped for another species, threatened loss of data. All options of the command lists *WinPro-h* option to the.

```
Host ~ $ winPRO h
```

Network

The network configuration in the host system must not be changed. The settings are made with the built-in DHCP server as the user mode network stack of QEMU. It is a virtual network (10.0.2.0) is generated with firewall and DHCP server (10.0.2.2). The DHCP server assigns the host system automatically assigns an IP address from 10.0.2.15. Possibly on the routing of which was provided by QEMU gateway (10.0.2.2) is defined. An access from the host system to the outside is now available. That is, if the host system has access to the Internet, has also the host system to automatically access the Internet. The firewall blocks all incoming connections to this virtual network. It is not possible to access from outside the virtual machine. The host system is protected. A test of the network connection with *ping* is not supported. Even browsing the Windows network is not possible. It IP addresses instead of names to be entered. Also, no applications to NT or 2000/2003-Server-Domains are possible. On the home directory in the host system are accessed via the share. The following URL in the file manager must be entered.

```
\ \ Host \ HOME
```

Directories outside of the home directory on the host system to be approved by symlinks. The following example allows access to the directory / tmp on the host system using the URL \\ Host \\ HOME \\ tmp

```
Host ~ $ ln -s / tmp $ HOME / tmp
```

Configuration

The configuration of virtual machines is done with variables. These are in the / etc / default / mergepro (global) and stored settings.local. The variables from the command line before the call to be set WinPro also. The memory of the host system is set with the variable MRGPRO_WINDOWS_RAMSIZE. The memory can be up to 2 / 3 of memory on the host system. The keyboard layout is defined with the variable MRGPRO_KBD_LANG. The size of the window is the variable MRGPRO_WINDOW_SIZE specified. Possible values are maximized and the information in pixels, for example 1024x768. The start time of the host system is reduced when the sound turned off, the variable MRGPRO_DISABLE_AUDIO with. Here are the configuration in the Bash shell.

```
Host ~ $ export MRGPRO_WINDOWS_RAMSIZE = 256 host = ~ $ export MRGPRO_KBD_LANG "de" host ~ $ export MRGPRO_WINDOW_SIZE = "maximized" host ~ $ export M
```

Be permanently variables in the settings.local in the list of virtual machine set them.

```
# WinPro / settings.local
# DO NOT DELETE OR CHANGE OR MRGPRO_WINDOWS_VERSION MRGPRO_GUEST_INSTALL_MODE
MRGPRO_WINDOWS_VERSION = "winxppro"
MRGPRO_GUEST_INSTALL_MODE = "shared"
# BEGINNING OF LOCAL USER SETTINGS
MRGPRO_WINDOW_TITLE = "WinPro"
MRGPRO_WINDOWS_RAMSIZE = 256
MRGPRO_KBD_LANG = "de"
MRGPRO_DISABLE_AUDIO = "yes"
MRGPRO_WINDOW_SIZE = "maximized"
```

System-wide user, these values from the root of the file / etc / default / mergepro stored in.

```
# Merge Pro defaults
MRGPRO_INST_DATE = "Fri February 31 2009 08:21:26 CET"
MRGPRO_NODENAME = "PCBSD"
MRGPRO_SYSNAME = "FreeBSD"
MRGPRO_SYSREL = "7.1-PRERELEASE"
MRGPRO_SYSVER = "FreeBSD 7.1-PRERELEASE"
MRGPRO_SYSMACHINE = "i386"
VARMERGE = "/usr/local/win4bsd"
STATICMERGE = "/usr/local/lib/win4bsd"
MRGPRO_SYSTEMTYPE = "bsd"
MRGPRO_MERGENAME = "Win4BSD Pro"
MRGPRO_CD_DEV_NAME = "/dev/acd0"
MRGPRO_CDA_SOCK_DIR = "/usr/local/win4bsd/cda"
MRGPRO_KBD_LANG = "de"
```

As user root optimizes the file / usr/local/lib/win4bsd/bin/mergepro-configs.sh. In the function add_to_count () is the variable COUNT = 1 to set.

```
add_to_count ()
{
    [$ QUIET-eq 0] && echo $ 1
    # COUNT = `expr $ COUNT + 1`
    COUNT = 1
}
```

In the host system, the following improvements are made. The System Restore should be disabled because the whole guest system in the host system can be secured. On Microsoft Windows XP is chosen to start menu, My Computer and click on System Information. In the System Restore tab to enable System Restore on all drives, disable. Printers under Microsoft Windows XP Start menu, printers and fax machines with printers to add configured. In the wizard choose Network printer and provides the connection to the printer \\ HOST \\ host-in printer. Then added for the driver installation to manufacturer Apple and the Apple LaserWriter printer in V23.0.

Often you need a standalone Windows application and not the entire desktop. This is configured in the host system with the following steps. You start the Registry Editor (Start menu Run: regedit) and check whether the necessary registry entries are in the.

```
Computer, HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, Windows NT, CurrentVersion, Winlogon
Userinit variable = B: \\ mrgrp32.exe
```

The log shall be made without login screen. On Microsoft Windows 2000 is in the Start menu, Settings, Control Panel, Users and Passwords must be user name and password to disable the user point. On Microsoft Windows XP is the Start Menu, Control Panel, user accounts, change the way users log on option to disable welcome screen use.

In the Registry Editor (Start menu, Run, type: regedit) the application is selected. In this example, it is Internet Explorer.

```
Computer, HKEY_LOCAL_MACHINE, SOFTWARE, Win4Lin:
Right-click, New, String: "SingleAppStart."
```

With a double application, the complete path to the value set as: C: Program Files \\ Internet Explorer \\ IEXPLORE.EXE. \\ Only after you restart Internet Explorer. With the option-desktop system that can host entire desktop still be started with the.

```
Host-deSKTOP ~ $ WinPro meinWin
```

International

The applied launch options can be viewed winpro.log in the file.

```
Host ~ $ cat ~ / WinPro / winpro.log
```

The image GUEST.IMG consists of a partition.

```
Host ~ $ fdisk WinPro / GUEST.IMG
```

Win4BSD based on the QEMU version 0.8.2.

```
Host ~ $ strings /usr/local/lib/win4bsd/bin/mergepro-core | \
grep "QEMU 0th"
QEMU 0.8.2
```

Win4Lin

Win4BSD is a commercial product from Virtual Bridges, Inc. and is based on QEMU. This means that the Microsoft Windows versions 2000 and XP or run individual Windows applications on Linux.

Win4Solaris

Win4Solaris is a commercial product from Virtual Bridges, Inc. based on QEMU. This means that the Microsoft Windows versions 2000 and XP or run individual Windows applications on Solaris (x86).

Other

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-

Images_anderer_Virtualisierungssoftware . Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools).

[<<< | # # | >>>](#)

"From http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_Win4 "

This page has been accessed 2920 times. This page was last updated on 27 September 2010 at 06:58 clock changed. Content is available under GNU Free Documentation License 1.2 .

Xen EISX Download Installation HOWTO domU

(Link to this page as [[QEMU-KVM-Buch / Notes / More virtualizer and emulators / Xen]])

<<< | # # # | >>> | English



Contents

- Xen 1
 - 1.1 EISX
 - 1.1.1 Installation
 - 1.1.2 Creating virtual machines
 - 1.2 Other

Xen

Website: <http://www.xen.org>

The software Xen is a hypervisor that is being developed at the University of Cambridge. Xen moves in as abstraction layer between the hardware and the guest systems and monitors the distribution of resources (CPU time, I / O cycles and so on). The Xen project uses source code from QEMU Project for the emulation of common PC components. The operational target of Xen, as with the Kernel-based Virtual Machine, the server consolidation. Xen dominated in contrast to the KVM paravirtualization. If the paravirtualization used, only matched host systems are used.

Xen is to install complicated and expensive. First, a boot loader is needed to start in several steps in the right order and the necessary software components. There must be a hypervisor and then the customized operating system for the privileged virtual machine to load. The description which is beyond the scope of a quick guide.

EISX

Website: <http://www.eisxen.org>

Download: <http://www.eisfair.org/home/download/>

The complicated configuration of Xen is on EISX not necessary. Based on the proven *Linux eisfair minimal distribution* to install a Xen server in minutes, and virtual machines can be easily set up themselves. The Linux distribution *eisfair* (Easy Internet Server - <http://www.eisfair.org/home/was-ist-eisfair/>) serves as a privileged virtual machine. EISX can also be used on old hardware.

Installation

EISX can QEMU or Kernel-based Virtual Machine are also tested (see http://qemu-buch.de/d/Gast-Systeme/_x86-Architektur/_Hypervisor). You restart the computer with the installation CD. First, ask if the entire hard disk to use *hda*.

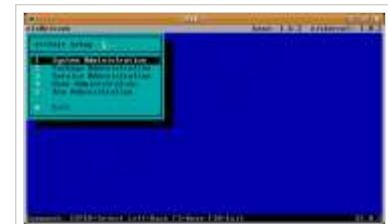
Use hda for installation (y / n) [yes] **[Enter]**

In test mode is no extra partition for the data necessary.

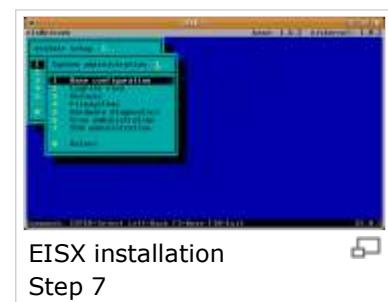
```
Create extra data partition / data (y / n)? N
Create partitions as listed above (y / n) y
```

After the security check will begin the installation. In the end, after the passwords for the *root* user and asked *ice*. Users of the *ice* is used to configure the system by using menus.

```
Setting root password for user:
Enter new UNIX password: *****
Setting password for user Ice:
Enter new UNIX password: *****
```



EISX installation
Step 6



EISX installation
Step 7



EISX
Creating a virtual machine
Step 1



EISX
Creating a virtual machine
Step 2



EISX
Creating a virtual machine
Step 3

In this example, the network configuration via DHCP.

```
Use DHCP for network card? (Y / n) [yes] [Enter]
```

After installation, the machine is rebooted without the CD. To configure the system log into *ice* as a user. It automatically starts the configuration program.

Creating virtual machines

can EISX In guest systems can be generated easily from templates. To create a host system from this template you select the menu item *Administration Xen, domU Administration, Create new domain from template*. The package includes templates for the Linux distribution are eisfair. For the first test you take over the guidelines:

```
eisfair1.tar.bz2
Name of domain [eisfair1]: [Enter]
Disk size in MB [512]: [Enter]
Memory size in MB [64]: [Enter]
Swap space in MB (0 = no swap) [128]: [Enter]
Power on when domain 0 boots (y / n) [yes]: [Enter]
```

It is the template select (*eisfair1.tar.bz2*). Then a name for the guest system is to forgive. More details are the size of the virtual disk, the RAM and the size of the swap area. EISX installed the new system and submit the generated image in the folder */data/xen/images/down*. Then the network is configured.

```
Please enter a host name [eisfair1]? Horst
Please enter an IP Address? 10.0.2.20
Please enter the subnet mask [255.255.255.0]: [Enter]
Please enter the default gateway? 10.0.2.2
Please enter the Doamin [lan.home]: [Enter]
Please enter the DNS? 10.0.2.3
```

For the guest system user credentials for the *root* obtained. Guest systems are the menu *Xen administration, domU administration, management domains* managed. This is the guest system (domain) input.

```
List Xen domains:
no order booted name
 10-50 no horst
Enter command ([1] to manage the domain, Enter = Return): 1
```

It will start the guest system.

```
Please enter command:
[B] oot,
b [o] ot & connect,
[N] ew boot order
[D] elete
Enter = Return: o
```

The guest system will boot and you will see the login screen. Will leave this console using the key combination **[Ctrl]+[]**. The operating system can be instant *eisfair* many software packages installed by (see <http://www.pack-eis.de>). It is possible to generate additional guest systems with the available templates. Furthermore, you can move their own templates. How EISX find on the Web site. A QEMU image with EISX and a template for generating virtual machines with Ubuntu 6.06 LTS server at the URL <http://www.oszoo.org/wiki/index.php/EisXEN-beta2a-with-ubuntu-6.06.1-server-template> to be downloaded.

Other

in Kernel-based Virtual Machine As with Xen is often the library *libvirt* as the basis for managing virtual machines used. A detailed description can be found at the URL <http://qemu-buch.de/d/Managementtools>

[/_libvirt-Tools](#) . A command summary is at the URL http://qemu-buch.de/d/Anhang/_libvirt listed.

If the host system is Ubuntu Linux, which is to create a virtual machine with *ubuntu-vm-builder* possible. A description can be found at the URL http://qemu-buch.de/d/Managementtools/_libvirt-Tools .

In order to create virtual machines with *openQRM* is first to install and activate the *plug-xen* (see http://qemu-buch.de/d/Managementtools/_OpenQRM). Then *host* is an appliance with the resource type to generate *Xen*. With the *Xen Manager* (Menu *Xen-plugin*) you submit to a new virtual machine.

How to convert virtual hard disks of different virtualizer and emulators can be found at the URL http://qemu-buch.de/d/Speichermedien/_Festplatten-Images_anderer_Virtualisierungssoftware . Some useful tools are explained in the Appendix (see http://qemu-buch.de/d/Anhang/_Nützliche_Tools). This brief overview is also used to get along with less well-known guest systems.

<<< | # # # | >>>

Retrieved from " http://qemu-buch.de/de/index.php/QEMU-KVM-Buch/_Anhang/_Weitere_Virtualisierer_und_Emulatoren/_Xen "

This page has been accessed 5933 times. This page was last updated on 27 September 2010 at 06:58 clock changed. Content is available under GNU Free Documentation License 1.2 .