„Kompatibilität“

Windows-Programme unter Linux laufen lassen und umgekehrt?

1 Allgemeines

„Linux ist nicht Windows“ (generelle Informationen und Einführung in die Thematik verschiedener Betriebssysteme, Technik und etwas Philosophie):

http://www.felix-schwarz.name/files/opensource/articles/Linux_ist_nicht_Windows/

Verschiedene Betriebssyteme funktionieren unterschiedlich bzw. verwenden unterschiedliche Technologien, daher laufen grundsätzlich Binärprogramme (Maschinencode) nur auf der Hard- und Software, für die sie ursprünglich gedacht sind.

2 Emulationen / APIs

„Emulatoren“ stellen einer Software ihre „gewohnte Umgebung“ zur Verfügung, so dass sie auf einem anderen Betriebssystem als dem, wofür sie gemacht ist, lauffähig wird.

Wine

(„Windows-Emulator“) stellt die Microsoft-Umgebung her, so dass Windows-Programme lauffähig werden, vorausgesetzt, es sind alle notwendigen „Windows-Komponenten“ installiert. Viele Programme erwarten ihre Laufzeitumgebung in Form von sogenannten „DLL“-Dateien.

Es gibt speziell für den Wine-Emulator Kompatibilitätslisten, die angeben, welche Windows-Software damit läuft.

(X)dos(emu)

Erlaubt das Starten von „DOS“-Programmen unter Linux (auch graphische DOS-Programme/Spiele), wobei im Hintergrund ein „echtes“ DOS als Open Source eingesetzt wird.

UAE

„Useless Amiga Emulator“, simuliert einen Commodore Amiga, und lässt die Amiga-Programme darauf laufen. Dies ist schon fast eine „Virtualisierung“, da der Amiga eine andere CPU und Architektur hat.

Spielkonsolen-Emulatoren

scummvm – Adventures

Für fast alle Spielekonsolen gibt es Emulatoren, die erlauben, die Spiele auch ohne entsprechende Spielekonsole unter Linux laufen zu lassen, allerdings ist dies nicht in jedem Fall legal (Stichwort „Kopierschutz“).

3 Virtuelle Maschinen

Virtuelle Maschinen, oder „Virtualisierung“, simulieren eine komplett andere Hardware, als im Rechner tatsächlich vorhanden ist, wobei die Maschinen-Anweisungen der Programme allerdings teilweise „nativ“ auf der echten CPU ausgeführt werden.

Hiermit werden NICHT einzelne Programme eines anderen Betriebssystems ausgeführt, sondern das „Gast-Betriebssystem“ tatsächlich vollständig gestartet. „Rechner im Rechner“.

Vmware (proprietär), mit graphischer Administrationsoberfläche,

Virtualbox (Open Source) , mit graphischer Administrationsoberfläche,

qemu/**kvm** (Open Open Source), Kommandozeilen-basiert, Oberfläche des Gast-Betriebssystems ist natürlich graphisch,

kvm ist fast identisch mit **qemu**, führt aber viele Befehle direkt auf der CPU bzw. der Host-Hardware aus, und ist deswegen schneller.

xen (Open Source), eine Virtualisierung, die im Linux-Kernel „eingebaut“ werden kann, und die fremde Betriebssysteme quasi als „Programm“ startet.

Vmware und **qemu** gibt es auch für Windows. Hiermit kann z.B. ein komplettes Linux-Betriebssystem mit Anwendungen in einem Fenster unter Windows laufen lassen, mit allen Linux-Anwendungen darin.

Der „simulierte“ Rechner hat, abhängig davon, ob die CPU „Hardware-Virtualisierung“ unterstützt, zwischen 40% bis fast 100% der Originalgeschwindigkeit des „echten“ Rechners.

4 Remote Services

Windows-Server / rdesktop

„**rdesktop**“ ist ein Client für Windows-Server über das „RDP“-Protokoll, und erlaubt es sich von Linux aus graphisch auf einem Windows-Server einzuloggen, und die dortige Windows-Oberfläche zu nutzen.

VNC Server und Client

„Virtual Network Computing“ (oder besser „Visual …“) ist ein Client-/Server-System, mit dem der Desktop auf andere Rechner „projeziert“ werden kann. Im „Viewonly“-Modus kann der entfernte Desktop nur „beobachtet“ werden, ansonsten darf auch der Desktop „bedient“ werden.

Beispiel:

Lehrer-Rechner (Server): **x11vnc -shared -viewonly**

Studenten-Rechner (Clients): **xvnc4viewer 10.0.20.15**

Die Clients können jetzt den Lehrer-Rechner „beobachten“, und sehen, was dort passiert, aber wegen „viewonly“ können sie in diesem Szenario keine Aktionen auf dem Lehrer-Rechner durchführen.

http://de.wikipedia.org/wiki/Virtual_Network_Computing

5 Kompatibilität vs. Plattformunabhängigkeit

Möglichkeit A: Die gleiche Software für verschiedene Betriebssysteme (und -versionen) anbieten.

Beispiele (Open Source): OpenOffice/LibreOffice, Firefox.

Beispiele (proprietäre Software): flash-Plugin, Java (Runtime) von Firma Sun, Opera-Browser, vmware-Virtualisierung).

Möglichkeit B: Die Software in einer betriebssystemunabhängigen Plattform herstellen (z.B. Java, Flash (proprietär), Browser-basiert mit Javascript/Ajax).

Voraussetzung dafür, dass es funktioniert, ist natürlich die jeweilige Laufzeit-Umgebung (Java-VM für Java-Programme, Flash für Flash-Programme usw).

6 Kommerzielles

Priorität kommerzieller proprietärer Software-Verkäufer:

Viele Kunden, die Lizenzen kaufen.

Beispiel: Spiele, die von Einzelpersonen gespielt werden.

Priorität kommerzieller Open-Source Softwareentwickler und -supporter:

Viele (oder wenige, aber dafür gut) zahlende Kunden, die Anpassungen und Support oder Dienstleistungen (Zugang) kaufen.

Beispiel: Online-Rollenspiele, bei denen weniger das Programm selbst, sondern mehr die Community-Erlebnis im Vordergrund steht.

In beiden Fällen ist gewünscht, dass möglichst viele die Software nutzen können, was die Motivation zur Interoperabilität erhöht.