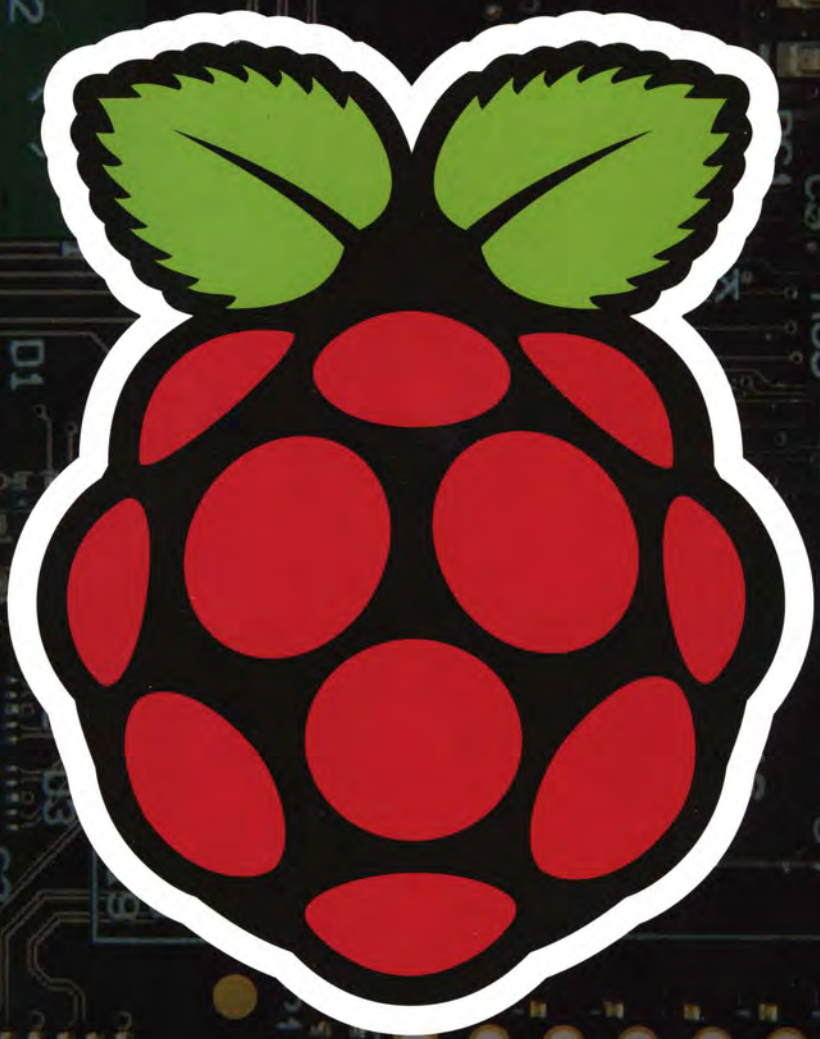
PC Games Hardware Guide präsentiert:

Raspberry Pi

Das Handbuch



ALLES ÜBER DEN 35-EURO-MINI-COMPUTER

Tipps | Tricks | Anleitungen | Projekte | Spaß

Grundlagen
Installation, Anschlüsse und Zubehör einfach erklärt

Tutorials
Experimente und Projekte zum Nachmachen

Coding
Sofort loslegen mit Scratch, vertiefen mit Python

Linux
Einführung und achtteiliger Lehrgang vom Profi

Hardware
Der Pi als Retro-Konsole oder Mediencenter

Basis-Wissen für Einsteiger und Fortgeschrittene

computec MEDIA EDITION

4 198667 509991 01

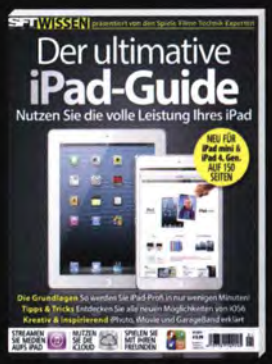
PCGH GUIDE 01/14
€ 9,99
Österreich 11,- Euro
Schweiz 17,- sfr
Benelux 11,50 Euro

PC Games Hardware Guide
Raspberry Pi: Das Handbuch
www.pcgameshardware.de

computec MEDIA EDITION

ALLE PREMIUM-BOOKAZINES IM ÜBERBLICK

25 JAHRE computec MEDIA – LEIDENSCHAFT FÜR GAMES



Bequem online bestellen:
shop.computec.de/edition

Jetzt auch für das iPad in der Computec-Kiosk-App

computec MEDIA KIOSK

computec MEDIA EDITION

PCGH Hardware Guide Raspberry Pi

computec MEDIA

Ein Unternehmen der MARQUARD MEDIA INTERNATIONAL AG
Verleger Jürg Marquard

Verlag Computec Media AG
Dr.-Mack-Straße 83, 90762 Fürth
Telefon: +49 911 2872-100
Telefax: +49 911 2872-200
redaktion@pcgameshardware.de
www.pcgameshardware.de | www.pcghx.de

Vorstand Albrecht Hengstenberg (Vorsitzender), Rainer Rosenbusch, Ingo Griebl

Chefredakteur PCGH GUIDE (V.i.S.d.P.) Christian Müller, verantwortlich für den redaktionellen Inhalt, Adresse siehe Verlagsanschrift
Redakteur für besondere Aufgaben PCGH GUIDE Lars Craemer
Layoutkoordination PCGH GUIDE Albert Kraus
Übersetzung, Layout MDV Textdienste

Chefredakteur PC Games Hardware Thilo Bayer (tb)
Product Manager Daniel Waadt (dw)

COO Hans Ippisch
Vertrieb, Abonnement Werner Spachmüller (Ltg.), Nikolaus Krier
Marketing Jeanette Haag
Produktion Martin Closmann (Ltg.), Jörg Gleichmar

www.pcgameshardware.de
Chefredakteur Online Thilo Bayer
Redaktion Andreas Link, Lars Craemer, Marco Albert, Frank Stöwer, Carsten Spille, Marc Sauter, Raffael Vötter, Stephan Wilke
Entwicklung Markus Wollny (Ltg.), René Giering, Aykut Arik, Tobias Hartlehnert
Webdesign Tony von Biedenfeld, Emanuel Popa

Anzeigen
CMS Media Services GmbH, Dr.-Mack-Straße 83, 90762 Fürth

Anzeigenleiter
Gunnar Obermeier, verantwortlich für den Anzeigenteil. Adresse siehe Verlagsanschrift

Anzeigenberatung Print:
Bernhard Nusser: Tel.: 0911-2872-254; bernhard.nusser@computec.de
René Behme: Tel.: 0911-2872-152; rene.behme@computec.de
Alto Mair: Tel.: 0911-2872-144; alto.mair@computec.de
Anne Müller: Tel.: 0911-2872-251; anne.mueller@computec.de

Anzeigenberatung Online:
freeXmedia GmbH
Ein Unternehmen der freenet Group
Deelbögenkamp 4c, 22297 Hamburg
Tel.: +49 (0) 40-513 06-650
Fax. +49 (0) 40-513 06-960
Mail: werbung@freeXmedia.de
www.freeXmedia.de

Anzeigendisposition: anzeigen@computec.de
Datenübertragung: via E-Mail: anzeigen@computec.de
Es gelten die Mediadaten Nr. 26 vom 01.01.2013

Abonnement - http://abo.pcgameshardware.de
Die Abwicklung (Rechnungsstellung, Zahlungsabwicklung und Versand) erfolgt über unser Partnerunternehmen DPV Direct GmbH:
Post-Adresse: Leserservice Computec, 20080 Hamburg, Deutschland

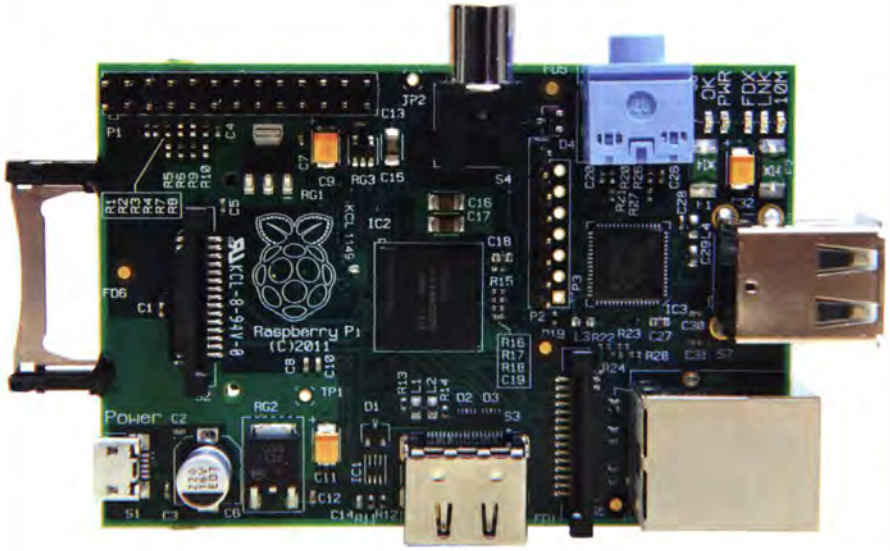
Ansprechpartner für Reklamationen ist Ihr Computec-Team unter:
Deutschland: E-Mail: computec@dpv.de, Tel.: 0911-99399098, Fax: 01805-8618002*
Support: Montag 07:00-20:00 Uhr, Dienstag-Freitag 07:30-20:00 Uhr, Samstag 09:00-14:00 Uhr
* (14 Cent/Min. aus dem dt. Festnetz, max. 42 Cent/Min. aus dem dt. Mobilfunk)

Österreich, Schweiz und weitere Länder: E-Mail: computec@dpv.de,
Tel: +49-911-99399098, Fax: +49-1805-8618002
Support: Montag 07:00-20:00 Uhr, Dienstag-Freitag 07:30-20:00 Uhr, Samstag 09:00-14:00 Uhr

Abonnementpreis für 12 Ausgaben: PCGH Magazin € 47,88 (€ 51,60 Österreich, € 59,88 Ausland), PCGH DVD € 60,- (€ 67,20 Österreich, € 72,- Ausland)

Einzelversand/Nachbestellung
Online: www.pcgh.de/go/shop, E-Mail: computec@dpv.de, Tel.: 0911-99399098 ; Fax: 01805-8618002*
* (0,14 €/Min. aus dem dt. Festnetz, max. 0,42 €/Min. aus dem dt. Mobilfunk)
Druck: RR Donnelley Europe, ul. Obroncow Modlina 11, 30-733 Krakau, Polen

Willkommen!

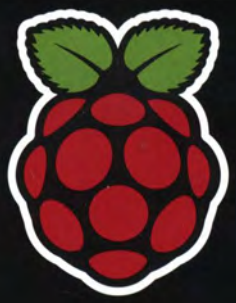


Viele Neuheiten im IT-Bereich werden heutzutage als revolutionär angepriesen, doch das ist häufig nur Marketing. Der Raspberry Pi ist indes tatsächlich revolutionär. Und damit meine ich gar nicht so sehr die Tatsache, dass ein Gerät, das nur aus einer einzigen bestückten Leiterplatte besteht, das ohne jeden Designanspruch daherkommt und hinter dem kein weltumspannender Konzern steht, sich praktisch aus dem Stand in anderthalb Jahren fast zwei Millionen Mal verkauft hat.

Vielmehr meine ich den Enthusiasmus und Erfindergeist, den dieser kleine Computer in einer Breite geweckt (oder wiedererweckt) hat, wie sie seit den seligen Heimcomputertagen der 1980er nicht mehr zu beobachten war und die man wirklich kaum anders bezeichnen kann als eine Revolution. Plötzlich beginnt sich die Öffentlichkeit wieder für das Programmieren am Rechner zu interessieren, und eine neue Generation junger Bastlerinnen und Tüftler scheint sich dessen bewusst zu werden, was die alten PC-Hasen noch wissen oder bereits vergessen hatten: dass man mit einem Computer weitaus mehr anstellen kann, als im Web zu surfen, Excel-Tabellen zu verwalten und Grand Theft Auto zu zocken. Und genau das hatten die Erfinder des Raspberry Pi – die übrigens in einer gemeinnützigen Stiftung organisiert sind und keinen Gewinn mit dem Produkt machen – auch im Sinn, als sie das Konzept erdachten.

Der Raspberry Pi hat die Abmessungen einer Scheckkarte und wird ohne Gehäuse geliefert, aber er ist ein voll funktionsfähiger Computer, und das zu einem unschlagbaren Preis von kaum mehr als drei Kinobesuchen. Das macht ihn zu einem idealen Experimentiergerät, und als solches wurde es auch von seinen vielen neuen Fans dankbar angenommen. Der Raspberry Pi eignet sich aber auch hervorragend als vollwertiger Rechner überall dort, wo Platz, Stromversorgung und finanzielle Mittel begrenzt sind. Raspberry Pis sind bereits ins Weltall und über den Ozean geschickt worden, sie dienen als Steuerungseinheiten von Robotern, als Sicherheitssysteme oder als Musikinstrumente – der Phantasie sind keine Grenzen gesetzt!

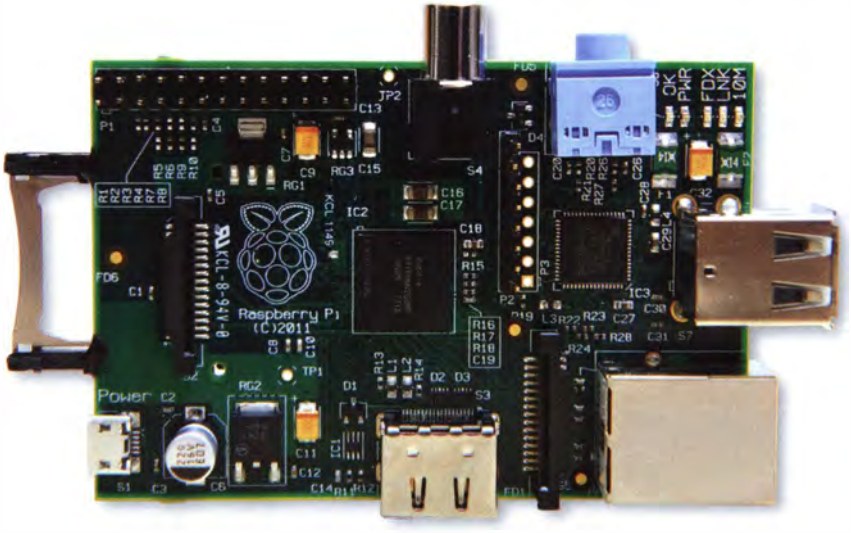
Lassen Sie sich vom Raspberry-Fieber anstecken, entdecken Sie die Welt des Programmierens, oder konstruieren Sie clevere Helferlein für alle möglichen Aufgaben. Willkommen in der spannenden, abenteuerlichen Welt des Raspberry Pi!



Inhalt

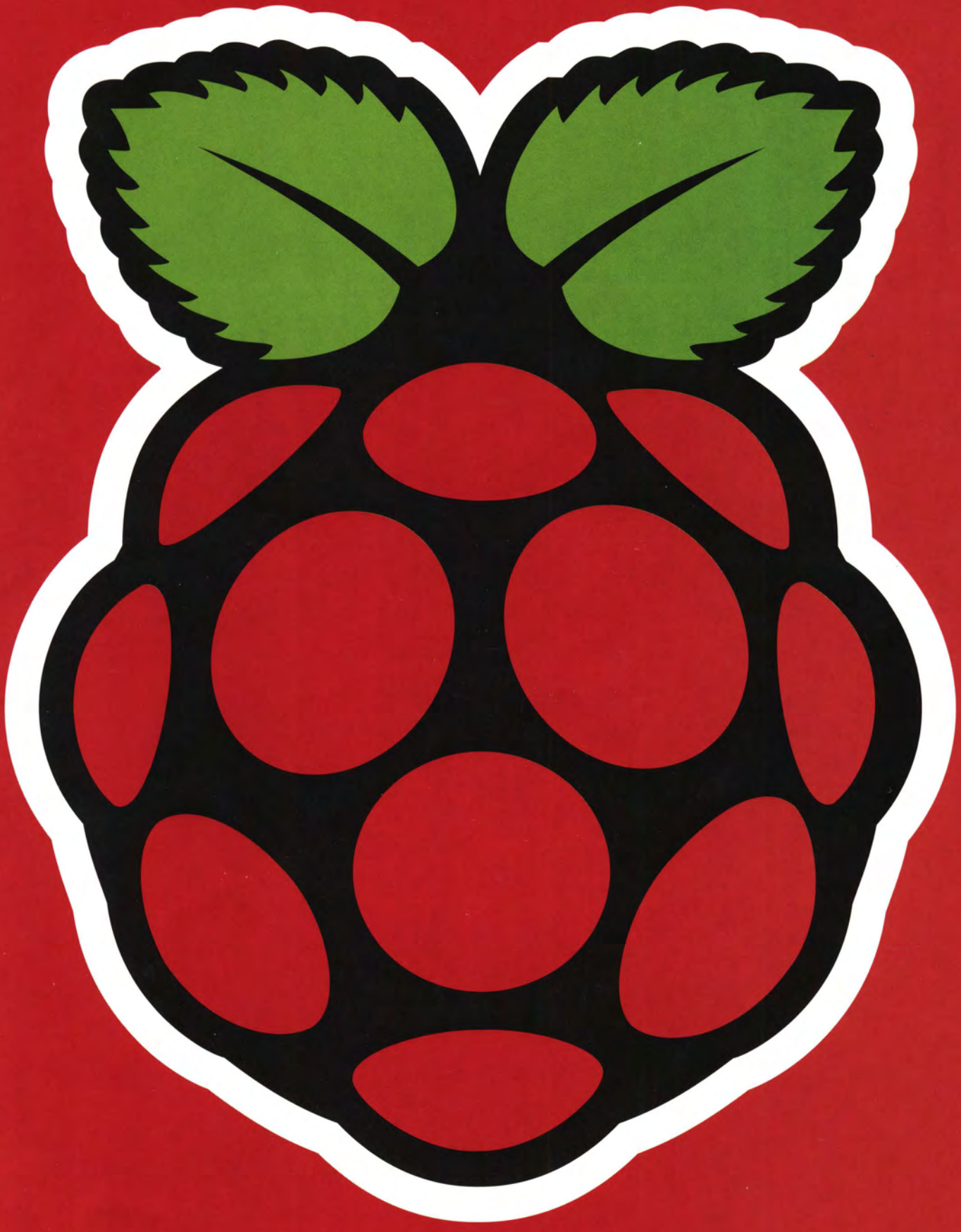
Grundlagen

Tutorials



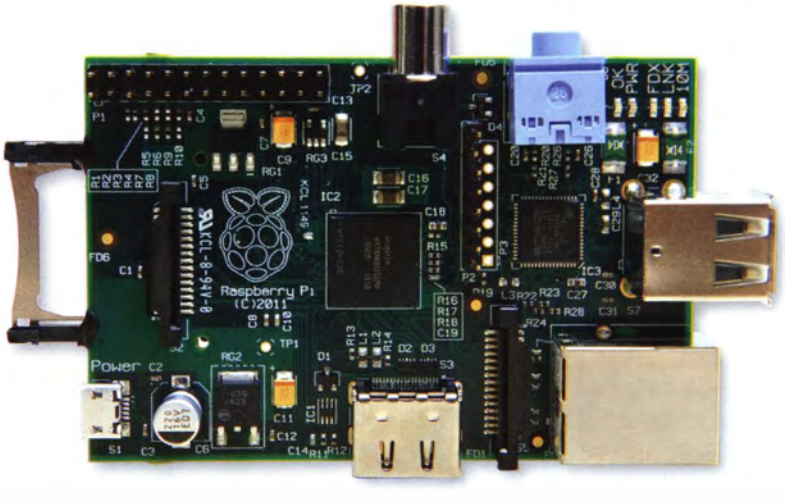
Programmieren

Linux



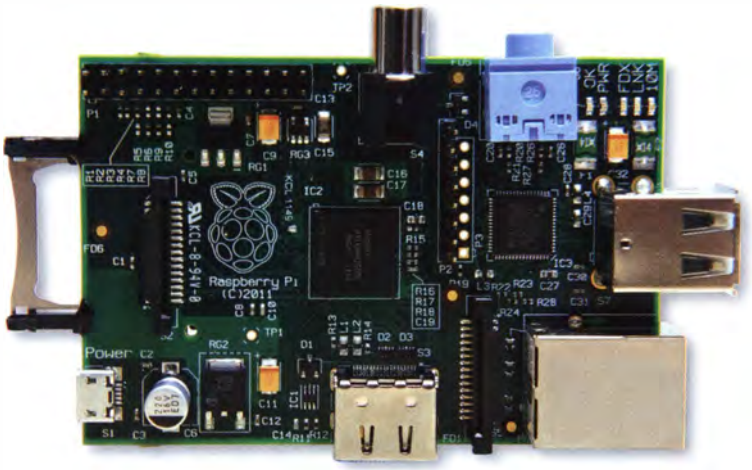
Grundlagen

Allerorten ist vom Raspberry Pi die Rede. Fast jeder Computer-Fan hat schon von der Platine mit dem Himbeerlogo gehört. – Doch was stellt man eigentlich mit dem Gerät an, wenn man es erst einmal ausgepackt hat? Die Beiträge in diesem Abschnitt geben Ihnen eine Einführung und einen ersten Überblick.



Ursprünge: Wie

Hier erfahren Sie etwas über die Entstehungsgeschichte des Raspberry Pi sowie über dessen einzelne Bauteile.



Heutzutage hält ein Computer seine Funktionsweise versteckt. Sie öffnen ein Dropdown-Menü, klicken mit der Maus auf ein Icon, und das Programm, das Sie gerade benutzen, macht das, was Sie wollen. Oder zumindest das, was es laut dem Ersteller der Software machen soll. Leider ist die Benutzung eines Computers für die meisten Menschen nur noch gleichbedeutend mit der Benutzung einer Software, die jemand anders für sie geschrieben hat, und nicht mehr mit der Entwicklung eigener Lösungen für eine Aufgabenstellung. Es stimmt natürlich, dass grafische Benutzeroberflächen den Computer einem weit größeren Kreis von Anwendern zugänglich gemacht haben, als dies ohne solche Programme je möglich gewesen wäre. Aber es stimmt genauso, dass die meisten Anwender nur noch eine grobe Ahnung davon haben, wozu ihr Computer eigentlich in der Lage wäre.

Doch nicht nur die Bequemlichkeit hält die Menschen davon ab, einen Blick unter die Haube ihrer Rechenmaschinen zu werfen, um herauszufinden, wie sie genau funktionieren. Ein weiterer Grund sind diverse Klauseln in den Endbenutzerverträgen von Windows und Mac OS X, die das Modifizieren des Programmcodes verbieten. Obwohl Sie eine ansehnliche Summe für das Betriebssystem bezahlt haben, ist es Ihnen nicht erlaubt, es zu „zerlegen“ und wieder „zusammenzusetzen“. Diese beiden Faktoren sorgen in Kombination dafür, dass die Anwender wenig Ahnung von den Funktionsabläufen innerhalb ihrer Computern haben.

Die Stunde des Raspberry Pi

Hier kommt nun Eben Upton ins Spiel, Studienleiter für Informatik am St. John's College in Cambridge. Im Rahmen seiner Tätigkeit ist er unter anderem am Zulassungsprozess für neue Studenten beteiligt. Dabei stellte er über die Jahre hinweg einen kontinuierlichen Verfall der Fähigkeiten bei den Bewerbern fest.

Je mehr Zeit Universitäten aufwenden müssen, um Bewerber mit unterdurchschnittlichem Vorwissen auf das Standardniveau zu heben, desto weniger Zeit bleibt, um den Studenten die wirklich interessanten Dinge beizubringen. Dies hat natürlich zur Folge, dass auch das Kenntnisniveau der Berufseinsteiger in der Informatikbranche sich mit der Zeit verschlechtert und Personalchefs es zunehmend schwerer haben, einen Bewerber mit ausreichender Qualifikation zu finden.

Um dem entgegenzuwirken, haben Eben Upton und einige Mitstreiter den Plan geschmiedet, ein Gerät zu entwickeln, das „hackbar“, billig und frei von Schutzansprüchen ist sowie zu einem Bruchteil der Kosten prinzipiell zu allem in der Lage ist, was auch ein normaler PC kann. Das Produkt, das die Gruppe aus dem Hut zauberte, war – na klar! – der Raspberry Pi.

Der BBC Micro

Der Raspberry Pi war nicht der erste Computer, der mit der Absicht entwickelt wurde, das IT-Wissen der britischen Bevölkerung zu verbessern. 1981 brachte Acorn Computers in Zusammenarbeit mit dem Computer-Kompetenz-Projekt der BBC den BBC Micro auf den Markt. Das Gerät war in zwei Versionen – Modell A und Modell B – erhältlich und wurde mit dem Zweck entworfen, darauf Programme laufen zu lassen, die Jugendliche im Umgang mit Computern trainieren sollten. Eine Fernsehsendung der BBC begleitete das Projekt. Der größte Unterschied zwischen den beiden Versionen war der Preis: 1981 kostete das Modell A 235 Pfund und das schnellere Modell B stolze 335 Pfund. Der BBC Micro wurde zu einem riesigen Erfolg und trug nicht wenig dazu bei, dass eine ganze Generation britischer Kinder mit einem soliden Wissen darüber aufwuchs, wie man einen Computer programmiert. Dass der Chip im Herzen des Raspberry Pi auf dem Design einer Firma – ARM – basiert, die aus Acorn Computers ausgegründet wurde, stellt dabei eine nette historische Fußnote dar.

› Zu heutigen Preisen würde das Modell B des BBC Micro umgerechnet 1.300 Euro kosten – der Pi ist dagegen für 35 Dollar zu haben.

der Pi entstand

Ein Blick auf den Raspberry Pi

Slot für die SD-Karte

Der Raspberry Pi hat keinen Speicher mit an Bord, weshalb Sie diesen mit einer SD-Karte nachrüsten müssen, um darauf das Betriebssystem und Ihre Dateien zu speichern. Die Idee dahinter ist, dass Sie nur das bezahlen müssen, was Sie brauchen: Das kann eine 4-GB-Karte für wenige Euro oder eine 128-GB-Karte für ein kleines Vermögen sein.

RCA-Video-Ausgang

Für ältere Fernseher gibt es auch einen RCI-Video-Anschluss. Falls Ihr Bildschirm keine Anschlüsse für RCA oder HDMI besitzt, gibt es Adapter auf den gängigen DVI-Anschluss.

3,5-mm-Audio-Klinke

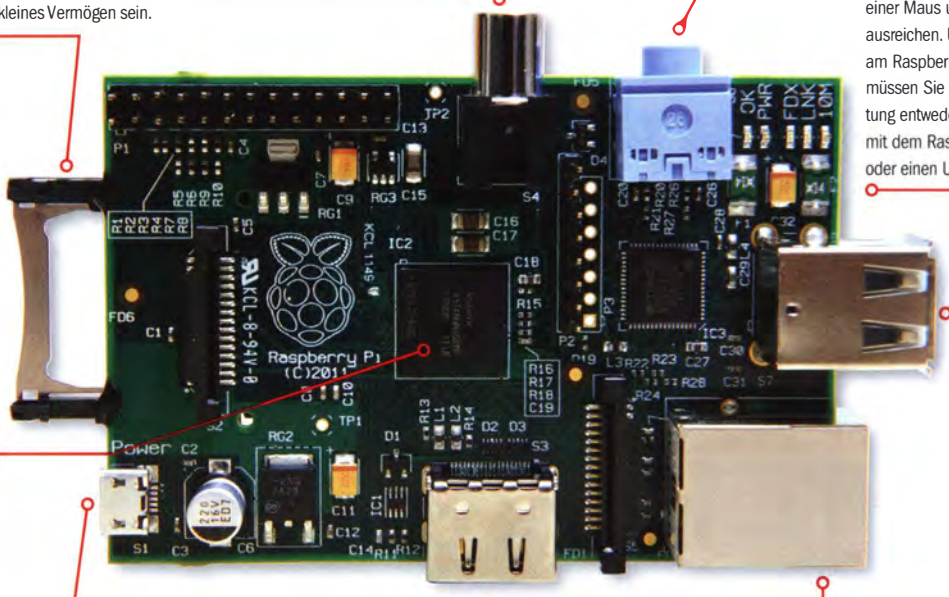
Mit diesem Anschluss verwandeln Sie den Raspberry Pi schnell in ein günstiges HiFi-System.

USB-Anschlüsse

Das Modell B hat zwei USB-Anschlüsse, die zum Betrieb einer Maus und einer Tastatur ausreichen. Um zusätzliche Geräte am Raspberry Pi zu betreiben, müssen Sie sich nach der Einrichtung entweder über das Netzwerk mit dem Raspberry Pi verbinden oder einen USB-Hub anschließen.

System-on-a-Chip

Dieses kleine Element ist das Herz des Raspberry Pi: Es besteht aus einem 700 MHz schnellen ARM11-Prozessor und einer Videocore-4-Grafikeinheit. Der Arbeitsspeicher sitzt im System-on-a-Chip: Modell A ist mit 256 MB RAM, Modell B mit 512 MB RAM ausgerüstet.



Stromanschluss

Das Raspberry Pi erhält seinen Strom über einen 5-V-Micro-USB-Stecker. So können Sie die Platine günstig mit Energie versorgen. Falls Sie ein Android-Smartphone besitzen, haben Sie das passende Netzteil wahrscheinlich schon zu Hause.

HDMI-Ausgang

Die Entwickler des Raspberry Pi gingen davon aus, dass die meisten Nutzer das Gerät an einem Fernseher oder einem Bildschirm verwenden würden: Deshalb gibt es einen HDMI-Anschluss, der den Raspberry Pi mit modernen TV-Geräten und Bildschirmen verbindet.

Ethernet

Stecker rein, und Sie sind mit dem Internet verbunden – es sei denn, Sie haben zum Modell A gegriffen, das keinen Ethernet-Anschluss besitzt. In diesem Fall verwenden Sie einen WLAN-Adapter, den Sie in einen USB-Anschluss einstöpseln.

Modell A oder Modell B?

Es gibt zwei Typen des Raspberry Pi, das Modell A und das Modell B. Bei beiden handelt es sich im Prinzip um den gleichen Computer. Die einzigen Unterschiede liegen im Preis, in der Größe des RAMs und in den vorhandenen Anschlüssen. Während das Modell B 512 MB aufweist, hat das Modell A nur 256 MB. Das Modell A verfügt außerdem nur über einen USB-Anschluss, während das Modell B derer zwei hat. Außerdem kommt das Modell B mit Ethernet-Anschluss, der dem Modell A komplett fehlt. Deshalb kostet das Modell B auch 35 Dollar, das Modell A ist dagegen schon für 25 Dollar zu haben. Das Modell B ist aufgrund dieser Unterschiede die vielseitigere Version der beiden und deshalb auch die Variante, die wir empfehlen sowie für dieses Heft verwendet haben. Falls Sie keinen Internetanschluss brauchen – vielleicht, weil es Ihnen um eine Möglichkeit geht, ein Embedded-System zu betreiben – ist das Modell A immer noch eine gute Wahl, die Ihnen hilft, die Kosten gering zu halten. Das gilt vor allem, wenn Sie viele davon brauchen. Trotzdem bevorzugen wir die Flexibilität des Modell B, die für wenige Euro zusätzlich zu haben ist.

Was ist: Linux?

Eine kurze Einführung in das Betriebssystem, das Ihren Blick auf den Computer nachhaltig verändern wird.

In der Anfangszeit der Computertechnologie mussten die Programmierer sämtliche Programme selber schreiben – nicht nur den Code, der ihre spezifische Fragestellung lösen sollte, sondern auch denjenigen, der eine Interaktion des jeweiligen Programms mit den Schaltkreisen des zu programmierenden Computers überhaupt erst möglich machte. Dies führte regelmäßig dazu, dass sich der Aufwand vervielfachte und viel Zeit abseits des eigentlichen Problems aufgebracht werden musste. Außerdem bedeutete es, dass nur solche Programmierer mit Computern arbeiten konnten, die ein Programmier-Interface zu erstellen in der Lage waren. Das wäre etwa so, als dürfte nur jemand den Führerschein machen, der weiß, wie man ein Auto baut. Eine enorme Einschränkung also.

Bereits zu Beginn der Computerrevolution war dieser Umstand sehr unpraktisch, aber vollends untragbar wurde er, als die Geräte schneller und komplizierter wurden. Es wurde immer offensichtlicher, dass eine Schnittstelle zwischen dem Metall des Computers und den Ideen des Benutzers gebraucht wurde: ein Vermittlungssystem, das alle notwendigen Anweisungen beherrscht, um den Computer voll zu nutzen, das aber gleichzeitig als abstrakte Basis für Dritt-Entwickler dient, damit sie funktionierenden Code schreiben können, ohne die baulichen Eigenschaften des Rechners zu kennen. Was benötigt wurde, war ein Betriebssystem.

Die heute dominierenden Betriebssysteme sind Windows und Mac OS X, nicht zuletzt dank des großen Marketingaufwands, den deren Hersteller Microsoft und Apple betreiben. Daneben gibt es jedoch noch viele andere Betriebssysteme.

Linux: allgegenwärtig

Eines der am weitesten verbreiteten Betriebssysteme – auch wenn es keine Werbespots dazu gibt – ist Linux. Linux ist das Betriebssystem, das Google und Facebook für ihre Server nutzen. Wenn die amerikanische Regierung ihre Arbeitsmarktzahlen erhebt, tut sie das mit Linux-Systemen. Linux arbeitet in Navigationsgeräten, in Set-Top-Boxen und in Mobiltelefonen (Android ist ebenfalls eine Form von Linux). Linux ist allgegenwärtig und treibt im Stillen unsere moderne Welt an. Sobald Sie gelernt haben, wie Linux auf dem Raspberry Pi funktioniert, werden Sie auch ein tiefes Verständnis der Computer erworben haben, die unser tägliches Leben ermöglichen.

Es gibt mehrere Gründe für die Allgegenwart von Linux. Einer der wichtigsten ist die Flexibilität des Betriebssystems. Während Mac OS X lediglich auf den etwa 20 von Apple produzierten Geräten funktioniert, kann Linux im Prinzip auf allem laufen – von den Superrechnern des CERN, die nach neuen subatomaren Partikeln suchen, über Computer, die vor 20 Jahren gebaut wurden, bis hin zu Ihrem kleinen Raspberry Pi: Linux funktioniert praktisch überall.

› *Bild rechte Seite:* **Seit 1996 ist der Pinguin namens Tux das offizielle Maskottchen von Linux.**

Linux: eine Lizenz, die alles erlaubt

Ein wichtiger Grund für die weite Verbreitung von Linux ist auch die Lizenz, unter der es vertrieben wird. Wenn Sie einen neuen PC kaufen, auf dem Windows installiert ist, müssen Sie vor der Verwendung des Betriebssystems einer Lizenzvereinbarung zustimmen. Es handelt sich dabei um einen „Ganz oder gar nicht"-Vertrag, der Sie dazu verpflichtet, die Software nicht zu kopieren oder zu modifizieren. Sogar die Anzahl der Geräte, auf denen Sie die Software installieren dürfen, ist beschränkt. Wenn Sie gerne einen Blick in den Quellcode werfen würden, um zu sehen, wie er funktioniert, haben Sie Pech, denn dies nicht zu tun, dazu haben Sie sich ebenfalls verpflichtet.

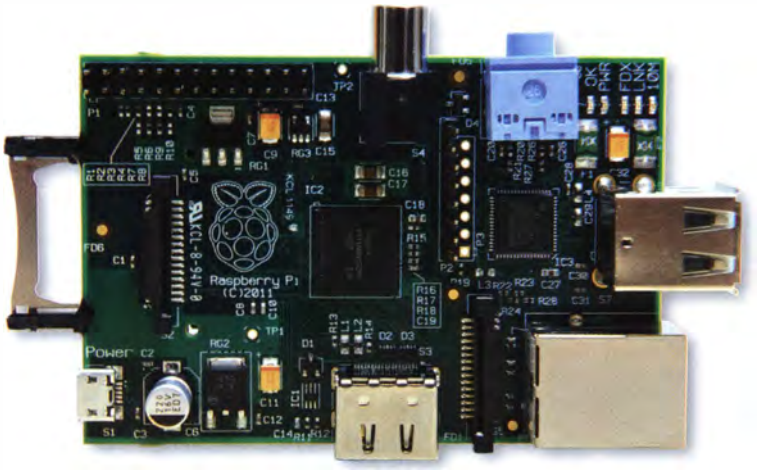
› **Die grafische Darstellung von Raspbian ist schlicht gehalten, damit sie auf der langsamen Hardware des Raspberry Pi gut funktioniert. Andere Versionen von Linux bieten aber auch 3D-Effekte.**

Linux ist im Gegensatz dazu komplett frei. Sie dürfen so viele Kopien anlegen, wie sie wünschen. Sie dürfen an der Software so viele Änderungen vornehmen, wie Sie möchten. Sie dürfen den Code herunterladen, sich damit selbst das Programmieren beibringen und alles nach Ihren eigenen Vorstellungen anpassen. Niemand wird bei alledem jemals an Ihre Tür klopfen und Sie der Verletzung von Urheberrechten beschuldigen. Die Linux-Lizenz ist so freizügig wie möglich gestaltet – das Einzige, das sie vorschreibt, ist, dass Sie im Falle einer von Ihnen durchgeführten Veränderung den neuen Code der Allgemeinheit zugänglich machen müssen. Ein mehr als fairer Deal in Anbetracht der Tatsache, dass Linux sowohl kommerziell als auch in puncto geistiges Eigentum vollkommen frei zur Verfügung steht.

Da es so weitreichende Optionen gibt, Linux zu modifizieren und damit herumzuspielen, gibt es buchstäblich Hunderte Varianten des Linux-Betriebssystems. Diese stammen von Weltkonzernen wie dem bereits erwähnten Google oder auch von Jugendlichen, die in ihrer Freizeit am Computer herumspielen. Es gibt diverse Linux-Versionen, die auf dem Raspberry Pi laufen, aber wir empfehlen eine ganz bestimmte, nämlich Raspbian. Diese Software basiert auf Debian, einer der ältesten Linux-Varianten, wurde aber für die Hardware des Raspberry Pi optimiert. Und nun kann das Abenteuer beginnen… ■

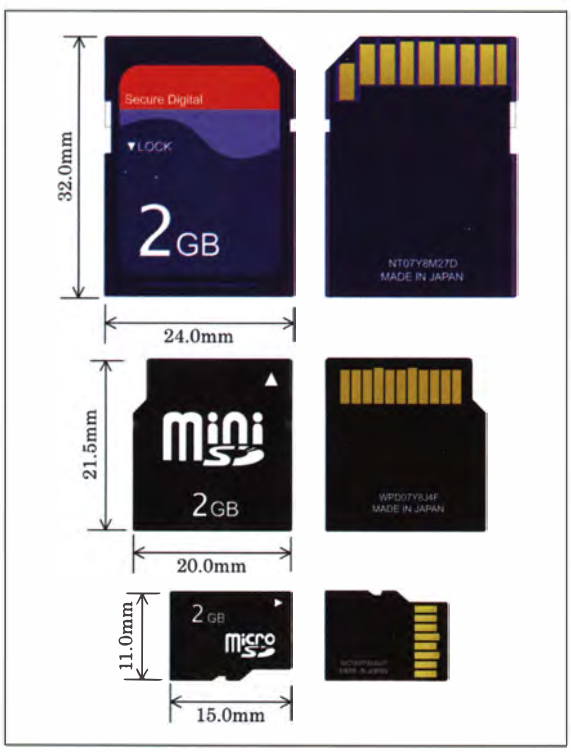
Richtig verbinden:

Unser detaillierter Guide, wie Sie das richtige Zubehör wählen und welche Verbindungen Sie für eine optimale Performance brauchen.



Bevor Sie mit dem Raspberry Pi loslegen können, müssen Sie einiges an Zubehör anschließen. Es lohnt sich, hier mit Sorgfalt vorzugehen, denn Probleme aufgrund falscher Verbindungen oder inkompatibler Peripherie sind im Nachhinein oft schwer aufzuspüren. Deswegen lohnt es sich nachzuschauen, welche Hardware am besten funktioniert – vor allem, wenn Sie den Kauf eines neuen Raspberry Pi planen.

Am besten fangen Sie mit der SD-Karte an, denn dies ist das vermutlich wichtigste Zubehör, das Sie für Ihren Raspberry Pi brauchen. Die kleinen Plättchen mit der abgeschnittenen Ecke dienen dazu, das Betriebssystem darauf zu speichern, und sind die einzigen Einheiten, von denen Ihr Raspberry Pi booten kann. Gerade zu Beginn müssen Speicherkarten einiges einstecken: Ihre SD-Karte muss willkürliche Neustarts sowie ständiges Entfernen und Einsetzen aushalten und ist ständig dem Risiko ausgesetzt, an eine unebene Platine angeschlossen zu werden.



Der Raspberry Pi kann wählerisch sein, wenn es um die Kompatibilität der Speicherkarte geht. Versichern Sie sich daher, dass Sie eine Karte kaufen, die funktioniert und passt. Achtung: Die Karte ganz oben hat die richtigen Maße!

Aus diesen Gründen ist es ein bisschen knifflig, die richtige SD-Karte auszusuchen. So ist es nicht empfehlenswert, eine Karte mit zu hoher Kapazität zu kaufen: Dabei bezahlen Sie einen hohen Preis für ein Stück Hardware, das schnell kaputtgehen kann. Andererseits ist eine bestimmte Menge an Speicher schon erforderlich, da Sie mindestens 2 GB benötigen, um das Standard-Betriebssystem Raspbian zu installieren. Dafür nicht genutzter Speicher kann dazu verwendet werden, Ihre eigenen Dateien oder Zusatzeinheiten zu speichern, darum sollten Sie auch wiederum nicht zu knapp kalkulieren.

Wir empfehlen Ihnen, für den Anfang eine Karte mit 4 GB Speicher zu kaufen und bei Bedarf ein zusätzliches USB-Gerät – etwa eine externe Festplatte – zu benutzen. Solche Geräte bieten Ihnen erstens mehr Speicher zu einem niedrigeren Preis, und zweitens bleiben so Ihre darauf gespeicherten Medien- oder Konfigurationsdaten erhalten, auch wenn Sie das Betriebssystem auf der SD-Karte überschreiben. Außerdem lohnt es sich, die Karte noch vor dem Kauf auf Kompatibilität zu überprüfen, da es mehrere SD-Karten gibt, die mit dem Raspberry Pi nicht funktionieren. Achten Sie des Weiteren darauf, eine SDHC-Karte zu kaufen („HC“ steht für „high capacity“, hohe Kapazität), da SDSC-, SDXC- sowie SDIO-Karten dafür bekannt sind, Probleme zu verursachen. Auf **http://elinux.org/RPi_SD_cards** werden sowohl die funktionierenden als auch die fehlerhaften Karten aufgelistet, basierend auf den Erfahrungen der Raspberry-Benutzer. Sogar Karten von großen Herstellern wie SanDisk oder Kingston versagen manchmal beim Pi ihren Dienst, daher lohnt es sich auf jeden Fall, vor dem Kauf die Kompatibilität zu überprüfen.

Falls Sie ein völliger Neuling sind, sollten Sie erwägen, sich eine SD-Karte mit vorinstalliertem Raspbian zu besorgen. Dadurch umgehen Sie sowohl das mögliche Problem der Inkompatibilität als auch die Schwierigkeiten beim Installieren des Systems auf der Karte. In der Praxis hatten wir bei unseren Versuchen selten Probleme mit Speicherkarten aus dem Fachhandel. Was Sie ebenfalls probieren können, ist die Verwendung einer alten SD-Karte von einer Kamera.

Energieversorgung

Man könnte annehmen, die Energiezufuhr für den Raspberry Pi wäre kein größeres Thema, aber das würde täuschen. Haben Sie nämlich nicht die richtige Versorgung für Ihre Pi-Konfiguration, dann können Sie nur schwer bestimmen, ob eventuell

Peripherie

Gehäuse für den Raspberry Pi

Obwohl dies nicht direkt die Leistung Ihres Raspberry Pi beeinflussen wird, lohnt sich die Anschaffung eines Gehäuses für die Platine. Standardmäßig ist der Pi sehr anfällig für Einwirkungen von außen, da die schützende Schicht sehr dünn ist und Metallflächen einen Kurzschluss verursachen und den Raspberry Pi beschädigen können. Dank der „Do it yourself"-Idee des Projekts gibt es für jedes Budget das passende Gehäuse. Wer sich die Ausgabe ganz sparen möchte, kann einen Gegenstand des täglichen Gebrauchs umfunktionieren, wie etwa die Hülle einer Audiokassette, oder sogar ein Gehäuse aus Lego-Steinen bauen – wenn Sie dabei durchsichtige Steine verwenden, können Sie weiterhin die Anzeigen die Status-LEDs sehen. Wenn Sie bereit sind, ein wenig Geld auszugeben, gibt es verschiedene bunte Acrylgehäuse für etwa 6 Euro. Viele dieser Gehäuse haben einen weiteren Vorteil: Sie können anmontiert werden. Nachdem Ihr Pi also einmal eingepackt ist, können Sie ihn überall anbringen – ob im Inneren eines Spieleautomaten oder an einem Regalbrett. Bedenken Sie dabei nur, dass der Pi bei hoher Belastung viel Wärme abgeben kann.

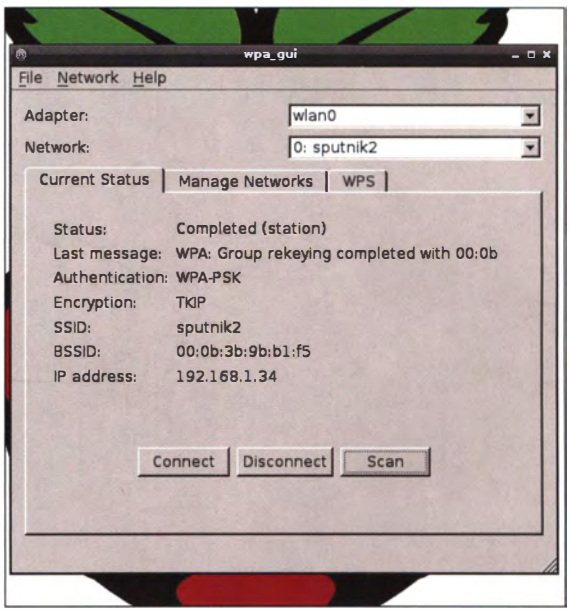
auftretende Probleme das Resultat der Energiezufuhr oder aber eines Softwarefehlers sind. Als wir beispielsweise am Tutorial zum Thema Fernsehaufzeichnung mit dem Pi gearbeitet haben, konnten wir mit unserer USB-Hardware stundenlang keinen einzigen TV-Kanal finden, obwohl alles vollkommen fehlerfrei aussah. Letztendlich war der von uns benutzte USB-Hub einfach nicht leistungsfähig genug. Um ausreichend Energie für Ihren Pi bereitzustellen, empfehlen wir Ihnen die ausschließliche Verwendung von Micro-USB-Ladegeräten mit mindestens 1,2 A. Mehr als 1,2 A verursachen keine Probleme, weniger kann zu unvorhersehbarem Verhalten führen. Solche Adapter liegen einerseits vielen modernen Mobiltelefonen bei, andererseits finden Sie sie in den meisten Elektrofachgeschäften. Jeder Adapter mit weniger als 1,2 A kann zu Problemen führen, wenn Ihr Pi stärker beansprucht wird.

Für externe USB-Verbindungen empfehlen wir dringend, einen guten USB-2.0-Hub (HiSpeed) zu verwenden. Diesen verbinden Sie mit Ihrem Raspberry Pi über einen Standard-B-Stecker und ein Standard-A-Kabel. Ältere 1.0-Hubs scheinen zwar zu funktionieren, doch hatten wir Probleme bei der Hardware-Kompatibilität – dieses Risiko sollte man lieber vermeiden. Außerdem müssen Sie sicherstellen, dass der Hub separat und nicht nur über den Raspberry Pi mit Strom versorgt wird. Bei vielen preisgünstigen Hubs gehört ein Netzteil nicht zum Lieferumfang, kann jedoch angeschlossen werden – achten Sie hierbei auf die jeweiligen Spezifikationen. Die Hubs müssen 5 V liefern und möglichst auch 3,5 A, damit Sie mehrere Geräte an den Pi anschließen können. Einfacher ist es, für ein wenig mehr Geld gleich einen Hub mit eigenem Netzteil zu erwerben. Achten Sie darauf, sämtliches Zubehör über den Hub und nicht direkt an den Pi anzuschließen, denn nur so profitieren Sie von der externen Stromversorgung.

› **Sie müssen sich überlegen, wie Sie sowohl das Raspberry Pi als auch die daran angeschlossenen USB-Geräte mit Strom versorgen möchten.**

Vernetzen

Für die Installation und Konfiguration müssen Sie den Raspberry Pi mit Ihrem Heimnetzwerk verbinden. Das erledigen Sie am besten über den Ethernet-Port auf der Platine, den großen RJ45-Anschluss an der Seite. Die meisten Internetverbindungen zu Hause finden über einen Wireless-Hub statt, der außerdem über einige RJ45-Anschlüsse verfügen sollte – falls Sie Geräte wie Spielkonsolen oder TV-Receiver per Kabel anschließen möchten. Der Raspberry Pi lässt sich auf die gleiche Art und Weise verbinden: Solange Ihr Hub in Betrieb ist, müssen Sie einfach beide Geräte per Ethernet-Kabel miteinander verbinden. Sollte das nicht möglich sein, können Sie eine andere Option nutzen, nämlich eine Verbindung via „Ethernet über Strom"-Adapter, auch als Powerline-Adapter bekannt. Einen der zwei Adapter verbinden Sie dabei mit einem freien Ethernet-Port Ihres »



› **Ethernet-Verbindungen sind einfach, aber viele USB-Geräte werden auch dann funktionieren, wenn Sie einfach das Desktop-WiFi-Programm benutzen.**

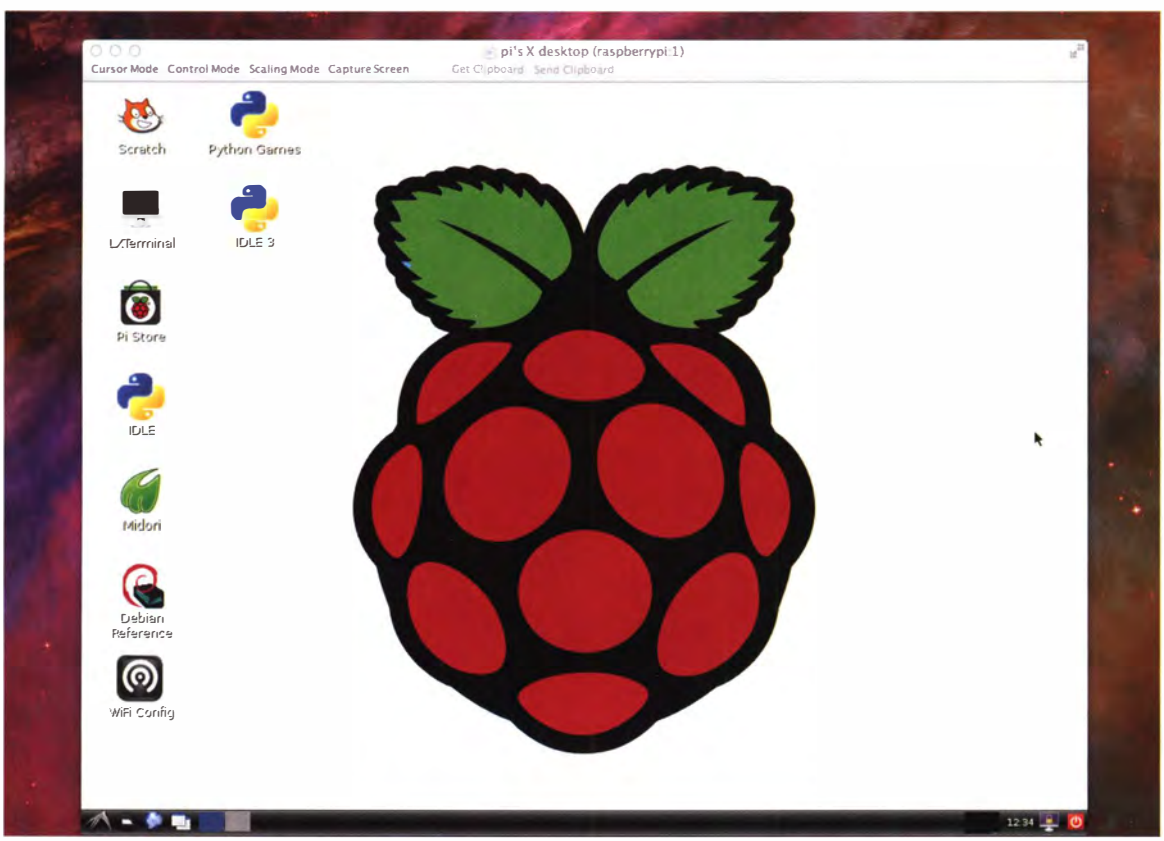
» Hubs und einer Steckdose, während der andere in eine Steckdose in der Nähe Ihres Pi gesteckt wird und über ein Ethernet-Kabel den Raspberry Pi mit dem Netzwerk verbindet.

Wenn Sie ein Modell A ohne Ethernet-Schnittstelle besitzen oder lieber eine kabellose Verbindung nutzen möchten, dann ist der Vorgang komplexer. Zuerst müssen Sie sich ein kabelloses USB-Gerät besorgen, das mit dem Raspberry Pi kompatibel ist. Wir empfehlen Ihnen, zuerst einen Blick auf das überprüfte Zubehör unter elinux.org zu werfen **(http://elinux.org/RPi_VerifiedPeripherals)** und ein Gerät zu wählen, das sicher funktioniert. Dadurch müssen Sie keine Treiber ohne Internetverbindung installieren. Außerdem sollten Sie das Gerät über einen USB-Hub mit eigener Stromversorgung verbinden und nicht direkt an den Pi anschließen, da kabellose Verbindungen beträchtliche Mengen Strom verbrauchen können. Das Einzige, das Sie danach noch zur Herstellung einer kabellosen Verbindung tun müssen, ist es, auf die Schaltfläche zur WiFi-Konfiguration auf dem Raspbian-Desktop zu klicken. Falls Ihr WiFi-Gerät richtig erkannt wird, sehen Sie „wlan0“ als Adapternamen und müssen nur noch auf die Scan-Schaltfläche klicken. Mit etwas Glück wird Ihr Heimnetzwerk erkannt und Sie müssen es nur noch doppelklicken, um das Konfigurationsfenster zu öffnen. Geben Sie das Netzwerk-Passwort ein und klicken Sie auf „Hinzufügen“. Schließen Sie das Fenster mit den Scan-Ergebnissen; die Statusseite sollte Ihnen anzeigen, dass Ihr Pi mit dem Netzwerk verbunden ist. Sie können die Netzwerk-Einstellungen jederzeit über die Netzwerkverwaltung ändern. Vor dem Schließen des Programms oder dem Neustart Ihres Raspberry Pi sollten Sie immer Ihre Konfiguration speichern.

› **HDMI lässt sich einfach in DVI konvertieren, um das Pi zusammen mit den meisten Flachbildschirmen zu benutzen.**

USB-Tastatur und -Maus

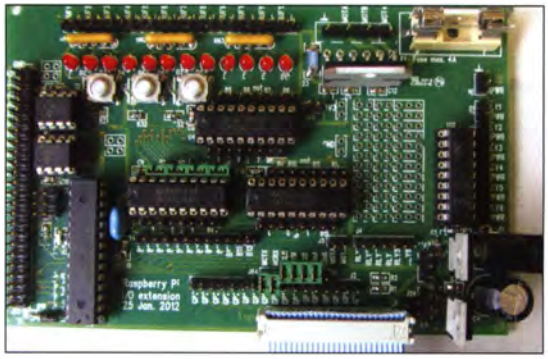
Wenn Sie über eine funktionierende Netzwerk-Verbindung verfügen, brauchen Sie nicht unbedingt eine Tastatur oder eine Maus. Ist das Raspbian-System installiert, können Sie sich per Fernsteuerung mithilfe eines Protokolls namens SSH mit Ihrem Pi verbinden. Installieren Sie über einen Windows-Rechner den kostenlosen Putty-Client. OS-X- und Linux-Nutzer verfügen über einen eingebauten SSH-Client: Tippen Sie **ssh**, gefolgt von **pi@** und der IP-Adresse Ihres Pi in die Kommandozeile (Ihr Router oder Hub hat eine Konfigurationsseite mit den IP-Adressen aller angeschlossenen Geräte). Das Passwort für den Standard-Account („pi“) ist „raspberry“. Wenn Sie verbunden sind, können Sie tippen und Ihren Raspberry Pi direkt konfigurieren, ohne eine Maus, Tastatur oder einen Bildschirm anschließen



› **Per Fernsteuerung können Sie Maus, Tastatur und Bildschirm eines anderen Geräts mit Ihrem Pi verwenden.**

Programmierbare Schnittstelle

Eine Anschlussart des Raspberry Pi unterscheidet sich deutlich von den anderen: Es handelt sich dabei um die Pins auf der Oberseite der Platine. Das sind die Allzweckeingabe- und Allzweckausgabe-Anschlüsse oder programmierbaren Schnittstellen (engl.: General Purpose Input Output Connectors, kurz GPIO), die etwa für spezielle Hardwareprojekte oder zur Verbindung mit Zubehör niedriger Stufe gedacht sind. Jeder Pin hat eine eigene Funktion und kann über die Entwicklungsumgebung oder mittels Programmiersprache gesteuert werden. Sie können die Pins etwa dafür benutzen, LEDs zu steuern oder Motoren, wenn der Pi die Steuereinheit eines Roboters bildet. Auf der Platine befinden sich 26 Pins (mit „P1“ gekennzeichnet), deren Funktion sich einhergehend mit der Überarbeitung des Pi leicht verändert hat. Daher müssen Sie wissen, welche Raspberry-Revision Sie haben – falls Sie etwa vorhaben, eine Tochterplatine zu kaufen, die direkt über die GPIO-Pins angeschlossen wird. Jeder Raspberry Pi, der Ende 2012 oder später hergestellt wurde, gehört zur Revision 2. Die bekannteste Erweiterungsplatine ist das Gertboard, das dem Pi alle möglichen Arten von programmierbarer Funktionalität einbringt – etwa Motorsteuerung, Digital-/Analog-Umwandler, zwölf LED-Leuchten oder auch neue Brücken und Anschlüsse. Allerdings benötigen Sie zu Beginn keine zusätzliche Platine, da die Pins auch ohne Modifikationen zu verschiedenen Zwecken genutzt werden können. Das neueste Update fügt der Platine eine P5-Stiftleiste direkt neben den P1-GPIO-Pins hinzu. Die acht Pins sorgen für mehr Energie sowie für vier zusätzliche GPIO-Funktionen, die vor allem für die Bereitstellung eines sekundären I²C-Kanals genutzt werden können; I²C ist ein serieller Datenbus.



› Sie können Platinen wie das Gertboard benutzen, die sich über die GPIO-Pins mit dem Raspberry Pi verbinden lassen.

zu müssen. Wenn Sie ein grafisches Interface haben wollen, können Sie auch ein Paket namens **tightvncserver** installieren und den Befehl **vncserver :1** ausführen, um einen ferngesteuerten Desktop zu starten. Benutzen Sie einfach einen beliebigen kostenlosen VNC-Client auf einem anderen Gerät, um sich mittels der IP-Adresse und des Ports 5901 Ihres Raspberry Pi zu verbinden. Danach können Sie Ihre Tastatur und Maus per Fernsteuerung nutzen.

Falls Sie jedoch eine Maus und eine Tastatur direkt an den Pi anschließen möchten, dürften Sie keine allzu großen Schwierigkeiten haben. Die meisten Modelle benutzen einen USB-Standard, weswegen Maus und Tastatur einfach funktionieren werden – es sei denn, Sie kaufen etwas Exotisches wie die neuesten kabellosen Microsoft- oder Logitech-Modelle. Sie müssen allerdings – wie bereits gesagt – beide Geräte über einen Hub mit externer Stromversorgung anschließen, sodass sie nicht zu viel Energie vom Raspberry ziehen. Falls Sie die Tastaturbelegung anpassen möchten, tippen Sie **sudo dpkg-reconfigure keyboard-configuration** in die Kommandozeile ein. Damit stellen Sie sicher, dass die Tasten Ihrer Tastatur mit denen beim Pi übereinstimmen.

> **„Falls Sie eine Tastatur anschließen möchten, dürften Sie keine großen Schwierigkeiten haben.“**

Display und Sound

Das Modell B des Raspberry Pi verfügt über zwei Anschlüsse, über die man Videosignale zu einem Display senden kann. Der gelbe Klinkenstecker dient zur Übertragung des zusammengesetzten Videosignals (Composite Video) und kann an einer großen Anzahl an Fernsehern angeschlossen werden, die normalerweise einen gelben Anschluss für externe Kameras oder Rekorder besitzen. Der gelbe Stecker befindet sich oft in der Nähe eines roten und eines weißen Steckers, die zur Audioübertragung (links/rechts) dienen. Versichern Sie sich, dass der HDMI-Port frei ist, falls Sie den Composite-Video-Anschluss nutzen wollen: Der Raspberry Pi stellt immer bevorzugt eine HDMI-Verbindung her, wenn er eine entdeckt, und schaltet erst auf Composite-Übertragung um, wenn er keine HDMI-Verbindung findet.

Allerdings ist der gelbe Stecker nicht die erste Wahl – es sei denn, Sie haben keine andere Option. Der moderne HDMI-Anschluss ist viel besser für die Übertragung geeignet, und Sie brauchen noch nicht einmal einen HDMI-Port an Ihrem Display, um ihn zu benutzen. Der Videoteil des Signals, der über HDMI übertragen wird, ist genau der gleiche wie das Videosignal, das etwa über das übliche DVI-Kabel eines PC-Monitors übertragen wird. Daher können Sie einfach ein Kabel mit einem männlichen DVI-Anschluss der Variante 24+1 auf der einen Seite und einem männlichen 19-Pin-HDMI-Stecker auf der anderen nutzen. Oder Sie wählen einen Adapter mit einem männlichen 19-Pin-HDMI-Stecker für den Raspberry Pi und einem weiblichen DVI-Anschluss für die Verbindung über ein übliches DVI-Kabel. Das alles mag jetzt sehr technisch und kompliziert klingen, doch handelt es sich sowohl beim DVI- als auch beim HDMI-Kabel und dem Adapter um übliche Teile, die Sie in jedem Elektronikgeschäft günstig besorgen können. Da die Kabel digitale Signale übertragen, lohnt sich die Investition in spezielle Adapter oder Umwandler nicht – es gibt kaum Anhaltspunkte dafür, dass solche Geräte die Signalstärke verbessern.

Ein weiterer Vorteil des HDMI-Anschlusses: Sie können darüber auch die digitalen Audiodateien von Ihrem Raspberry Pi übertragen. Ihr Fernseher oder Verstärker muss dieses Feature unterstützen, damit es funktioniert, und Sie werden ein bisschen an der Pi-Konfiguration herumspielen müssen, aber es ist eine gute Lösung, wenn Sie Ihren Raspberry als Medienhub benutzen wollen.

Ein HDMI-zu-DVI-Umwandler bringt Ihnen bei der Soundübertragung nichts, da die Soundkomponente bei der Umwandlung weggelassen wird, aber Sie haben immer noch den analogen Audioanschluss zur Verfügung. Dessen Ausgabe ist zwar mit Nebengeräuschen verbunden, aber immerhin gut genug, um Videos zu einem Fernseher zu streamen. ■

Betriebssystem:

Mit Windows-Tools kopieren Sie Raspbian einfach auf eine SD-Karte.

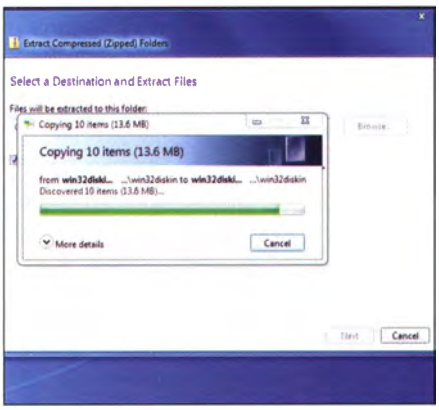
Schritt-für-Schritt-Anleitung: Windows

1 Tools herunterladen

Obwohl Windows von Microsoft nicht viel mit der Linux-Version, die auf dem Raspberry Pi läuft, gemein hat, bedeutet dies nicht, dass Windows-Anwender im Nachteil wären, wenn es um die Einrichtung eines funktionierenden Pi-Systems geht. Das liegt unter anderem daran, dass die vielen verschiedenen Einsatzzwecke von Windows und die zahlreichen Tools im Prinzip gar nicht so unterschiedlich zum Ansatz des Raspberry Pi sind. Beide Systeme bieten dem Nutzer grundsätzlich die gleichen Freiheiten bei der Verwendung von Anwendungen. Wenn man genauer sucht, findet man sogar bei Windows eine Kommandozeile, und viele Open-Source-Programme sind auch für den Betrieb mit Windows ausgelegt.

Windows-Anwender profitieren darüber hinaus von einer relativ gefahrlosen Installationsroutine in Form der Open-Source-Software *Win32 Disk Imager*. Als Erstes sollten Sie sich daher das Tool unter folgender Adresse herunterladen: **sourceforge.net/projects/win32diskimager**

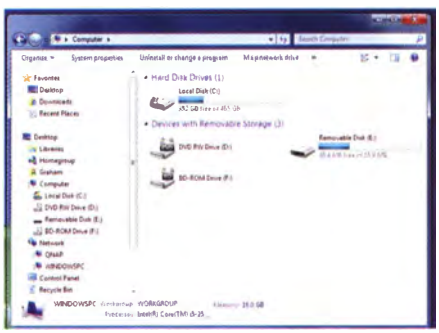
Sie brauchen *Win32 Disk Imager* nicht zu installieren, es reicht aus, wenn Sie das ZIP-File mit einem Rechtsklick in einen Ordner entpacken. Um Raspbian auf eine SD-Karte zu kopieren, benötigen Sie natürlich auch eine Kopie der entsprechenden Image-Datei. Unsere Methode funktioniert grundsätzlich mit allen Raspberry-Pi-Betriebssystemen und kann auch für das Schreiben anderer Images auf ein externes USB-Gerät verwendet werden.



› **Sie können das Open-Source-Tool *Win32 Disk Imager* für den Schreibvorgang verwenden.**

2 USB-Speicher prüfen

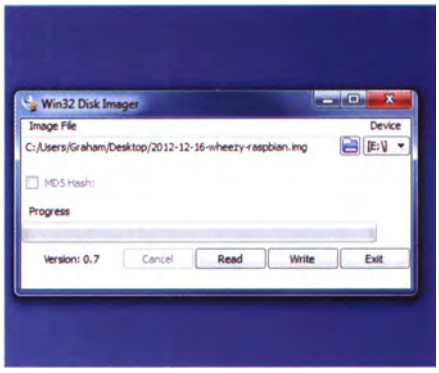
Bevor Sie das Tool zum Schreiben des Image-Files starten, sollten Sie sichergehen, dass auf der SD-Karte keine wichtigen Dateien mehr gespeichert sind und der SD-Kartenspeicher einen selbsterklärenden Namen wie beispielsweise „Raspbian“ trägt. Viele PCs besitzen einen eingebauten SD-Kartenleser, allerdings gibt es Berichte, nach denen es mit solchen Kartenlesern öfter Probleme geben soll, wenn man eine bootbare Version von Raspbian auf einer SD-Karte erstellen will. Die besten Ergebnisse konnten wir mit einen Kartenleser erzielen, der extern via USB mit dem PC verbunden war. Sollten Sie ebenfalls auf Probleme stoßen, sollten Sie als Erstes beim Kartenleser ansetzen. Die SD-Karte sollte nach dem Einschieben in den Kartenleser als eigenes Laufwerk angezeigt werden – merken Sie sich den Laufwerksbuchstaben, und prüfen Sie bei dieser Gelegenheit nochmals, ob das Laufwerk noch wichtige Daten enthält.



› **Windows zeigt in der Übersicht die Größe der Partitionen, nicht der Laufwerke an.**

3 Win32 Disk Imager starten

Jetzt können Sie die ausführbare Datei des Open-Source-Tools *Win32 Disk Imager* starten. Manchmal warnt Windows vor unbekannten Dateiquellen, im Zweifelsfall können Sie die ausführbare Datei von einem aktuellen Virenprogramm prüfen lassen. Seien Sie sich bei Windows-Dateien lieber ganz sicher, denn unter Linux gibt es kaum Viren – also besser am Anfang etwas mehr Sorgfalt walten lassen, als sich der Gefahr einer Virusinfektion auszusetzen. Das Hauptfenster des Tools ist nicht besonders intuitiv, für den Moment sind aber nur zwei Schaltflächen für die erfolgreiche Arbeit wichtig. Ein Knopf ist zuständig für die Auswahl des Image-Files, der andere – rechts daneben – erlaubt die Zielauswahl des Ortes, auf den die Daten des Image-Files geschrieben werden sollen. Wenn Sie auf das kleine Ordner-Icon klicken, müssen Sie dem Programm zeigen, wo das Image-File liegt. In der Regel werden nur *.img-Dateien angezeigt. Der kleine „Geräte“-Knopf zeigt dem Programm im Anschluss den Ort, wohin die Dateien geschrieben werden sollen. *Win32 Disk Imager* wählt meist das korrekte Ziel automatisch aus, aber prüfen Sie unbedingt den Laufwerksbuchstaben, sonst werden eventuell wichtige Dateien unbeabsichtigt gelöscht.

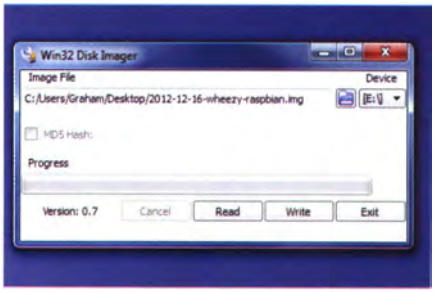


› **Das User-Interface ist zwar schlicht, aber es zeigt alle wichtigen Schaltflächen.**

Raspbian

4 Daten schreiben

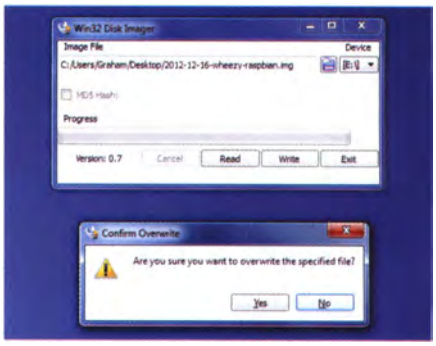
Wenn Sie alles geprüft und korrekt eingestellt haben, können Sie auf den „Write“-Knopf drücken. Es erscheint eine Warnung, die erneut die Gefahren für das Ziellaufwerk aufführt, diese können Sie allerdings ignorieren. Die SD-Karte wird auf dem untersten „Level“ mit den neuen Daten überschrieben, dazu zählen auch die Partitionstabelle und weitere grundlegende Formatierungen. Diese Low-Level-Formatierung ist notwendig, damit die SD-Karte später auch bootbar ist. Nach dem Start sollte, falls vorhanden, eine kleine LED beim SD-Kartenleser flackern, so können Sie ebenfalls sicherstellen, dass Sie den richtigen Laufwerksbuchstaben ausgewählt haben. Man kann nicht oft genug betonen, dass Sie hier extrem sorgfältig zu Werke gehen müssen, denn so ein Schreibvorgang auf einer der internen Festplatten sorgt dafür, dass alle Daten auf diesem Datenträger verloren sind. Der Schreibvorgang benötigt rund 20 Minuten, dieser wird von einer Balkenanzeige und von der Angabe der Schreibleistung pro Sekunde ergänzt.



› Während die Daten auf die SD-Karte geschrieben werden, informiert Sie ein Balken über den Fortschritt.

5 Die Installation prüfen

Bevor Sie die neu beschriebene SD-Karte in Ihren Raspberry Pi stecken, sollten Sie die Installationsdateien prüfen. Dieser Schritt gibt Ihnen auch einige Einblicke in die neue Formatierung der SD-Karte. Schauen Sie im ersten Schritt mithilfe der Laufwerksübersicht auf der SD-Karte nach, ob die neuen Dateien tatsächlich auf dem Speichermedium vorhanden sind. Die neue Linux-Partition auf der SD-Karte kann aber von einem Windows-PC nicht gelesen werden, da Windows das Linux-Dateisystem nicht versteht. Nutzen Sie zur Überprüfung daher ein gesondertes Windows-Tool, welches Sie durch die Suche nach „Erstellen und Formatieren einer Festplattenpartition“ finden können. Auf der Übersichtsseite der Computerverwaltung werden alle Laufwerke und Partitionen, die derzeit angeschlossen sind, angezeigt. Suchen Sie den Laufwerksbuchstaben der SD-Karte, darunter sollten drei Partitionen angezeigt werden: eine im FAT-Dateisystem mit rund 56 MB Speicherplatz, dann die Linux-Partition sowie eine dritte, die den nicht zugewiesenen Speicherplatz anzeigt. Wenn Sie also drei Partitionen auf der SD-Karte vorfinden, ist dem Anschein nach alles so verlaufen, wie es geplant war.



› Die Partitionsanzeige von Windows liefert brauchbare Hinweise, ob der Kopiervorgang ohne Probleme abgelaufen ist.

NOOBS – es geht noch einfacher!

Es ist grundsätzlich immer von Vorteil, wenn man die einzelnen Abläufe und Maßnahmen kennt, die beispielsweise zur Erstellung einer bootbaren SD-Karte erforderlich sind. Daher geben wir Ihnen für Windows, OS X und Linux auch diese Schritt-für-Schritt-Anleitungen an die Hand. Es gibt allerdings eine noch einfachere Methode, mit der man das gewünschte Resultat erzielen kann.

Dazu benötigen Sie die Software NOOBS, mit der man Raspbian und andere Linux-Versionen auf einem Raspberry Pi installieren kann. NOOBS ist ein Recovery-Tool für Raspberry Pi und enthält Images von OS-Varianten wie Raspbian, RiscOS, Arch Linux, Pidora, Raspbmc und OpenElec.

Das ganze Paket ist rund 1 GB groß und unter **www.raspberrypi.org/downloads** erhältlich. Um NOOBS zu nutzen, benötigen Sie eine SD-Karte mit 4 GB oder mehr Speicherplatz, als Dateisystem muss FAT32 verwendet werden. NOOBS wird installiert, indem Sie den Inhalt des NOOBS-Zip-Files auf die leere SD-Karte kopieren. Nun können Sie bereits die SD-Karte in einen Raspberry Pi stecken und das System starten. Beim ersten Bootvorgang wird NOOBS automatisch geladen und begrüßt Sie mit einem sehr übersichtlichen Aufbau. Um ein Betriebssystem zu installieren, müssen Sie im zugehörigen Menü einfach nur „Install OS“ auswählen – fertig. Es folgt ein grafischer Installationsassistent, der Sie über das ausgewählte Betriebssystem informiert. Eine der besten NOOBS-Funktionen ist, dass Sie beim Booten einfach die Shift-Taste gedrückt halten können und dann ein anderes der zur Verfügung stehenden Betriebssysteme auswählen können.

Beachten Sie aber, dass der Start mit einem anderen OS die Daten des vorherigen löscht.

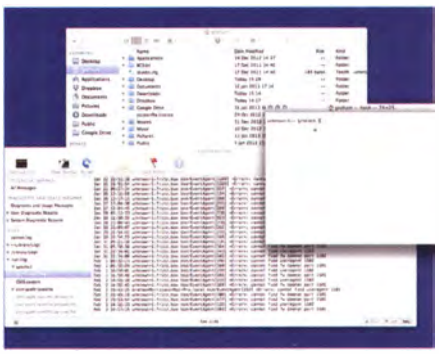
Betriebssystem:

Für Apple-Anwender gibt es eine schnelle Installationsmöglichkeit ohne großen Aufwand.

Schritt-für-Schritt-Anleitung: Apple

1 Die Vorbereitung

Obwohl OS X an jeder Stelle die Apple-Handschrift trägt und viele Dinge stark vereinfacht implementiert sind, hat das Apple-OS doch viele Dinge mit Linux gemeinsam. Beide Betriebssysteme basieren auf einem älteren Multi-User-System namens UNIX, und viele Tools und Anwendungen haben denselben Ursprung. Dies gibt erfahrenen OS X-Anwendern einen Vorteil, denn wenn Sie bereits mit der Kommandozeile von OS X gearbeitet haben, wird Sie das Linux-System vor keine großen Herausforderungen stellen. Ebenfalls sehr ähnlich sind die Konzepte der Anwenderkonten, der Verzeichnisse, des Dateiaustauschs und der Netzwerk-Drucker. Sie können den Raspberry Pi mit OS X über die Kommandozeile konfigurieren und steuern, ohne dabei spezielle Software installieren zu müssen. Darüber hinaus sind auch virtuelle Desktops kein Problem, und in dieser Disziplin können beide Systeme ihre Wurzeln nicht verleugnen. Bevor Sie allerdings diese Schnittstellen des Mac nutzen können, müssen Sie Raspbian natürlich erst auf einer Speicherkarte installieren.

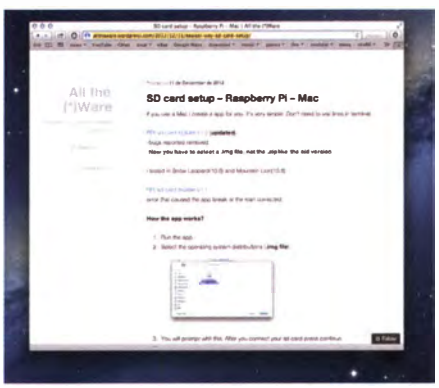


› **OS X bietet von Haus aus Tools, die Sie auch auf einem Raspberry Pi finden.**

2 Tools herunterladen

Die Standard-Installationsvariante von Raspbian mit einem Mac ist, genau wie bei Linux, über die Kommandozeile durchzuführen. Über diese installieren Sie Raspbian auf einer SD-Karte. Wenn Sie diesen Weg gehen möchten, folgen Sie der Schritt-für-Schritt-Anleitung für Linux und passen Sie die Gerätenamen denen bei OS X an. Es gibt auf dem Mac aber auch eine einfachere und sicherere Lösung, bei der die Installation über ein grafisches Benutzerinterface angegangen wird. Dazu benötigen Sie das Tool *RPi-sd card builder*, welches derzeit in der Version 1.2 verfügbar ist. Das Tool finden Sie unter: **alltheware.wordpress.com/2012/12/11/easiest-way-sd-card-setup.**

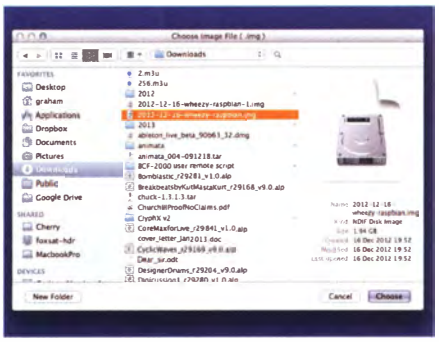
Speichern Sie das Tool in einem lokalen Ordner auf Ihrem Mac, und stellen Sie sicher, dass Sie die aktuelle Version des Raspbian-Images zur Hand haben, da dieses bereits im ersten Schritt der Installation benötigt wird. Die aktuelle Version des Raspbian-Images finden Sie unter **www.raspberrypi.org/downloads**. Verwenden Sie die Version, die mit dem Namenszusatz „wheezy“ versehen ist. Da die Image-Datei nur rund 500 MB groß ist, sollte der Download nur wenige Minuten dauern – dies hängt natürlich auch von Ihrem Internetanschluss und der Server-Geschwindigkeit ab.



› **Der *RPi-sd card builder* erspart Ihnen den mühsamen Weg über die Kommandozeile und ist für Einsteiger ideal.**

3 Den Builder starten

Der Download ist als Zip-File verpackt. Auf einem Mac werden die Zip-Dateien in der Regel nach dem Download automatisch entpackt. Sie erhalten dann eine Datei mit einem Namen wie **2012-12-16-wheezy-raspbian.img** – wobei die genaue Bezeichnung vom Download-Datum und der jeweiligen Software-Version abhängt. Wenn Sie die Datei selbst entpacken müssen, genügt ein Doppelklick auf die heruntergeladene Zip-Datei. Der *RPi-sd card builder* sollte nach dem Herunterladen ebenfalls im Download-Ordner abgelegt worden sein und ist recht einfach durch das Raspberry-Pi-Icon zu identifizieren. Starten Sie nun den *RPi-sd card builder*, und dieser fordert Sie auf, das jeweilige Image auszuwählen, das Sie für die Installation auf Ihrem Raspberry Pi heruntergeladen haben – da Sie das Image vorher in einem speziellen Ordner abgelegt haben, sollte es leicht zu finden sein. Anschließend drücken Sie noch auf den „Choose“-Knopf, und das Tool beginnt mit der Verarbeitung der Raspbian-Image-Datei.



› **Der *RPi-sd card builder* fordert Sie auf, manuell den Speicherort des Image-Files für die Installation auszuwählen.**

Raspbian

4 SD-Karte auswählen

Der *RPi-sd card builder* wird sich nach dem Start beschweren, dass keine SD-Karte gefunden wird – egal, ob diese korrekt angeschlossen ist oder nicht. Wir verwenden für unsere Arbeiten einen Standard-USB-Kartenleser, ein ähnliches Gerät sollten Sie mit der eingelegten SD-Karte an Ihrem Mac anschließen. Stellen Sie sicher, dass der Schreibschutz der SD-Karte nicht aktiviert ist. Bei Karten mit diesem Feature finden Sie an der Seite einen kleinen Plastikschieber – stellen Sie sicher, dass dieser korrekt eingestellt ist. Achten Sie ebenfalls darauf, dass auf der eingelegten SD-Karte keine Daten mehr gespeichert sind, die Sie behalten möchten, denn diese werden im folgenden Prozess vollständig überschrieben.

Der angeschlossene SD-Kartenleser wird automatisch nach dem Anschluss auf dem Desktop angezeigt. Nachdem Sie den Kartenleser samt Karte angeschlossen haben, klicken Sie in dem Programm *RPi-sd card builder* auf „Continue“. Der folgende Bildschirm zeigt alle Laufwerke an, hier müssen Sie den SD-Kartenleser und die SD-Karte auswählen – seien Sie 100 % sicher, dass Sie das richtige Ziel auswählen, denn sonst wird im schlimmsten Fall Ihre Festplatte gelöscht. In der rechten Spalte der Übersicht sehen Sie den Namen des ausgewählten Geräts, dieser sollte mit dem Namen der SD-Karte übereinstimmen. Vergleichen Sie auch die Größenangabe der SD-Karte mit Ihren Informationen. Wenn alles passt, klicken Sie auf „OK“.

› **Überprüfen Sie den Namen des Zielgeräts zur Not dreimal, damit Sie absolut sicher sind.**

5 Raspbian schreiben

Nun werden Sie gebeten, Ihr Passwort einzugeben. Sie müssen am Mac als Administrator angemeldet sein – eine weitere Ähnlichkeit mit dem Linux-System. Die meisten Mac-Anwender haben ohnehin Administratorenrechte, also können Sie ein gewünschtes Passwort in der Regel ohne Vorarbeiten eingeben. Ein weiteres Fenster verkündet nun, dass das Gerät erst entfernt („unmount“) werden muss. Dies ist erforderlich, da Inhalte im „Low Level“-Format geschrieben werden müssen und dies am Mac nicht möglich ist, wenn das Gerät vom Desktop aus erreicht werden kann. Warten Sie also, bis das Icon der SD-Karte vom Desktop verschwunden ist, dann klicken Sie auf „Continue“. Der eigentliche Schreibvorgang dauert rund 15 Minuten. Hinter dem rotierenden Zahnrad oben am Bildschirm verbirgt sich zusätzlich eine Statusleiste.

› **Sie müssen Ihr Passwort eingeben, um der Anwendung die benötigen Schreibrechte einzuräumen.**

6 Die Installation prüfen

Wenn der Vorgang abgeschlossen ist, sollte die SD-Karte für den ersten Einsatz im Raspberry Pi bereit sein. Bevor Sie sie benutzen, sollten Sie sich den Inhalt der SD-Karte aber nochmal auf dem Desktop anschauen. Dadurch können Sie schon früh Fehler bei der Vorbereitung ausschließen, damit es später keine Probleme gibt. Kurz nach dem Abschluss des Kopiervorgangs sollte die SD-Karte wieder auf dem Desktop auftauchen. Dieses Laufwerk sollte nur 18 MB Daten beherbergen, und angeblich sind nur noch etwa 41 MB auf dem Laufwerk frei. Dies liegt daran, dass Sie sich so zwar die Boot-Partition anzeigen lassen können, aber die anderen Partitionen auf der SD-Karte für den Mac unsichtbar sind. Wenn Sie diese Partitionen sehen wollen, müssen Sie die Datenträgerverwaltung des Mac verwenden. Diese zeigt bei der Auswahl das jeweilige Speichermedium, und wenn Sie es auswählen, können Sie sich auch die Partitionen anzeigen lassen. Wenn alles korrekt abgelaufen ist, sehen Sie zwei Partitionen und etwas ungenutzten Platz auf der SD-Karte. Diesen können Sie später mit einem Linux-Tool dem freien Patz auf der SD-Karte hinzufügen.

› **Prüfen Sie mit der Datenträgerverwaltung die Installation auf der SD-Karte, bevor Sie den Pi starten.**

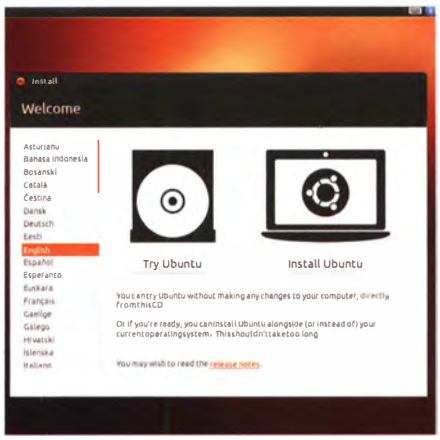
Betriebssystem:

Die Installation von Linux auf dem Raspberry Pi mithilfe eines Linux-Rechners ist vermutlich die sicherste Variante, und etwas lernen können Sie dabei auch noch.

Schritt-für-Schritt-Anleitung: Linux

1 Funktionsvielfalt

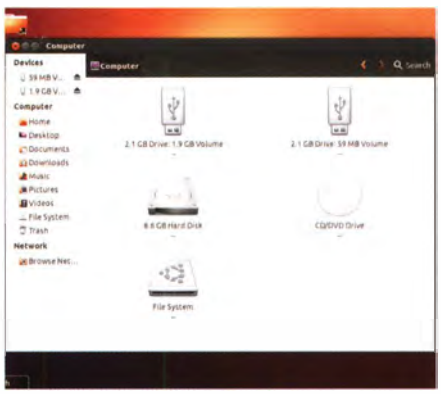
Wenn Sie bislang Linux noch nicht verwendet haben, müssen Sie sich deswegen keine Sorgen machen. Linux ist mittlerweile so einfach zu verwenden wie andere Betriebssysteme auch. Bei manchen Dingen ist es sogar einfacher zu bedienen. Man muss beispielsweise keine Treiber suchen, und Anwendungen werden quasi über einen eigenen App Store installiert. Und da Sie ohnehin Linux auf einem Raspberry Pi installieren wollen, können Sie auch schon mit den Vorbereitungen unter Linux beginnen. Wir empfehlen Ubuntu für die ersten Schritte mit Linux, die Tipps funktionieren im Prinzip aber auch mit jeder anderen Linux-Distribution. Linux ist auch für Notfälle eine gute Option, denn es kann von einer CD gestartet werden – ganz ohne Festplatte. Nach dem Einlegen einer Ubuntu-CD erscheint ein Auswahlmenü. Wählen Sie hier das Optionsmenü „Try Ubuntu“, denn so kommen wir auf den Ubuntu-Desktop, den Sie für Notfälle benötigen. Die Software ist kostenlos im Netz erhältlich und kann auf CD gebrannt werden.



› **Auch ohne Linux installiert zu haben, können Sie mit Ubuntu das System bis zum Desktop laden.**

2 Vorbereitung

Allerdings ist das Problem mit dem Start von der Ubuntu-Boot-CD, dass Sie damit Raspbian nicht herunterladen können, da Sie die Dateien nicht speichern können. Der Grund liegt darin, dass der Ubuntu-Anwendung zu wenig Speicher für den gesamten Download von Raspbian zur Verfügung steht. Sie müssen die Daten also auf einem externen Datenspeicher ablegen, aber nicht auf der SD-Karte, auf der ja das Image erzeugt werden soll. Wenn Sie ein installiertes Linux-System besitzen, sieht die Sache natürlich anders aus, dann können Sie die Daten einfach auf dem Datenträger ablegen. Nachdem das Image heruntergeladen wurde, sollten Sie sich ein Bild von der SD-Karte machen. Alle Daten auf dem Datenträger werden gelöscht. Stellen Sie also sicher, dass keine wichtigen Informationen auf der SD-Karte abgelegt sind.



› **Wenn Sie unter Ubuntu ein USB-Laufwerk einlegen, zeigt Ihnen Ubuntu sämtliche Partitionen auf diesem an.**

3 ImageWriter installieren

Wir werden das Tool *ImageWriter* als grafisches Frontend für den Schreibvorgang von Raspbian nutzen. Dieses Tool kann über das Software-Center von Ubuntu installiert werden. Dieses starten Sie durch einen Klick auf das Korb-Icon. Suchen Sie dann nach „imagewriter“. Es sollte nur ein einziges Tool angezeigt werden. Machen Sie einen Doppelklick auf das Programm, und es erscheint eine Anzeige, nach der die Software von der „universe“-Quelle erhältlich ist. Diese Quelle ist standardmäßig nicht aktiviert, sondern muss mit einem Klick auf „Use this Source“ freigegeben werden. Warten Sie, bis der Fortschrittsbalken durchgelaufen ist und die interne Angebotsliste aktualisiert wurde. Löschen Sie das Suchfeld, und geben Sie erneut „imagewriter“ ein. Das Softwareangebot sollte nun aktualisiert worden sein, und Sie können nun auf einen „Install“-Knopf klicken. Sie werden sich vielleicht fragen, warum man ein Zusatzpaket, nicht aber Raspbian ohne zusätzlichen Speicherplatz installieren kann. Das liegt daran, dass das Paket deutlich kleiner als Raspbian ausfällt.

› ***ImageWriter* kann auch mit der Live-CD von Ubuntu installiert und genutzt werden.**

Raspbian

4 Das Raspbian-Image schreiben

ImageWriter muss mit der „gemounteten“ SD-Karte gestartet werden, sonst startet es nicht und beschwert sich, dass kein Speicherplatz verfügbar ist. Wenn der Hauptbildschirm erscheint, müssen Sie der Software einige Parameter verraten. Der erste ist die Position des Raspbian-Images, das Sie auf den USB-Stick schreiben wollen, der zweite bezeichnet das Gerät, auf welches das Image geschrieben werden soll. Die zweite Einstellung ist besonders wichtig, da Sie, falls Sie mehrere Speicherplätze auf Ihrem System haben, sicherstellen müssen, dass Sie die Daten nicht auf das falsche Laufwerk aufspielen, um nicht unabsichtlich irgendwelche Daten zu löschen. Es wird auch der Name des Geräteherstellers angezeigt, dies könnte Sie ebenfalls vor einer falschen Auswahl bewahren. Nun können Sie das Programm mit einem Klick auf „Write to Device“ starten. Wenn das USB-Laufwerk mit der SD-Karte eine LED für Laufwerksaktivitäten besitzt, sollte diese jetzt flackern. Der Schreibprozess wird eine gute Weile dauern, die genaue Zeitspanne hängt von der Geschwindigkeit der USB-Ports und anderer Komponenten ab. Normalerweise können Sie von rund 15 Minuten ausgehen, danach können Sie die SD-Karte testen.

› ***ImageWriter*** **benötigt nur das Image und die SD-Karte als Ziel, dann kann es auch schon losgehen.**

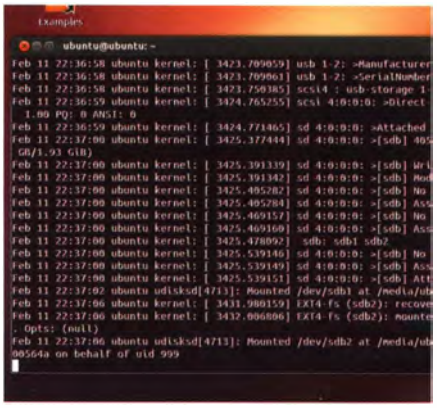
5 SD-Karte testen

Anders als Windows und OS X kann Linux als einziges Betriebssystem die beiden Partitionen der neuen Raspberry-Pi-SD-Karte lesen. Die erste ist in einem Windows-FAT-Format formatiert und sollte rund 60 MB groß sein. Über diese Partition startet das System, und anschließend übergibt Raspbian quasi die Kontrolle an die zweite Partition. Diese ist mit den wichtigsten Inhalten von Raspbian angefüllt, hier ist also das Root-Dateisystem von Linux zu finden. Beide Partitionen werden erkannt, wenn Sie die SD-Karte an ein Linux-System anschließen. Die Linux-Partition sieht der Ihres Ubuntu-Systems recht ähnlich. Dies liegt daran, dass beide Systeme vom selben Ursprung namens Debian abstammen. Der „Home“-Ordner besitzt beispielsweise einen Anwender-Ordner, in dem Dateien und Settings gespeichert werden. Raspbian ist auf einen einzigen Anwender „pi“ vorkonfiguriert. Dies lässt sich später natürlich schnell ändern. Sie können sehen, was sich in diesem Ordner befindet, wenn Sie auf „Home“ klicken. Wenn Sie fertig sind, „unmounten“ Sie die SD-Karte und stecken Sie diese in Ihren Raspberry Pi.

› **Linux kann das Windows- und Mac-Dateisystem ohne die Installation von Zusatzsoftware problemlos auslesen.**

6 Alternativ-Installation

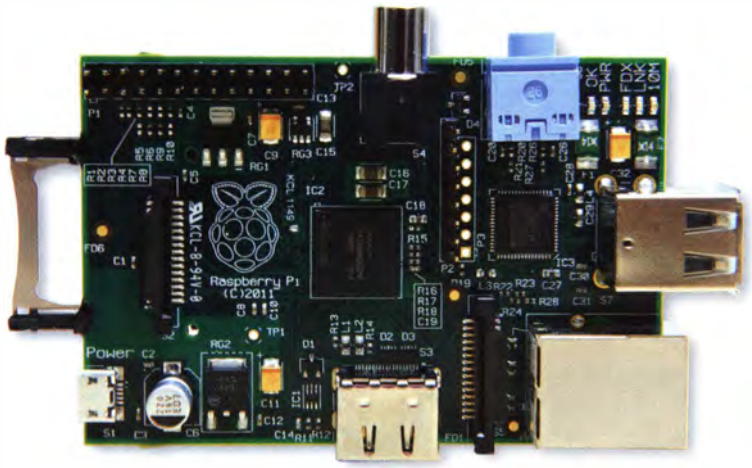
Es gibt eine weitere Methode, Raspbian auf eine SD-Karte zu schreiben. Da diese allerdings relativ gefährlich für den Inhalt anderer Datenträger sein kann, sollten Sie diesen Weg nur gehen, wenn die anderen nicht verfügbar sind. Bei dieser Methode müssen Sie über die Kommandozeile arbeiten und das **dd**-Kommando verwenden. Dieser Befehl kopiert die Daten quasi „roh“, also Byte für Byte. Wenn Sie sich hier mit der Zieleingabe vertun, wird Ihre wichtigste Festplatte schlimmstenfalls mit einem unbrauchbaren Raspbian-Image überschrieben. Nehmen Sie die SD-Karte aus dem System und starten Sie das Terminal von Ubuntu. Dieses ermöglicht nun die Kommandozeileneingabe. Tippen Sie **tail -f /var/log/syslog** ein, und stecken Sie die SD-Karte in den Rechner. Halten Sie nach einem Systemevent mit der Schreibweise **sdb: sdb1** Ausschau. Dies bedeutet, dass eine neue Festplatte erkannt wurde, in diesem Fall die eingelegte SD-Karte. Es werden vermutlich auch viele weitere Ausgaben angezeigt, da das Linux-System versucht, alle Inhalte auf der SD-Karte zu erfassen. Falls es „gemountet“ ist, sollten Sie es jetzt „unmounten“ und dann den Befehl **sudo dd bs=1M if=raspbian.img of=/dev/sdX** eingeben. Sie müssen jedoch den ***.img**-Dateinamen durch die korrekte Beschriftung des Image-Files ersetzen und den **/dev/sdX**-Node um Ihre eigenen Spezifikationen ergänzen. Das Image wird jetzt auf die SD-Karte geschrieben, ganz ohne Desktop-Befehle.



› **Sie können den dd-Befehl durch die Tastenkombination STRG+C abbrechen.**

Fehlerbehebung:

In den allermeisten Fällen werden Sie diese beiden Seiten gar nicht benötigen – Ihr Pi wird einfach starten. Falls es dennoch nicht klappen sollte, finden Sie hier einige Hinweise zur Fehlerbehebung.



Der große Unterschied zwischen Ihrem Raspberry Pi und einem herkömmlichen PC ist die Tatsache, dass das Pi kein BIOS besitzt. Das BIOS ist das Erste, was Sie beim Start Ihres PCs sehen. In einem teilweise an vergangene DOS-Zeiten erinnernden Bildschirm werden Ihnen Informationen über Speicher, CPU und angeschlossene Festplatten angezeigt, allerdings in einem kaum lesbaren Tempo. Dennoch spielt das BIOS eine große Rolle bei der Suche nach Fehlern wie etwa einer schadhaften CPU. Ein modernes BIOS meldet solche Probleme nicht nur via Bildschirmtext, sondern auch über akustische Signale und blinkende LEDs auf dem Motherboard. Ohne das BIOS fehlt Ihnen beim Raspberry Pi also ein wichtiges Werkzeug zum Erkennen von vermeintlichen Fehlern. Auch wenn Ihr Pi beim ersten Mal ohne Weiteres booten wird, so steigt die Wahrscheinlichkeit eines Problems während eines Startvorgangs, je mehr Sie an der Funktionalität Ihres Geräts basteln. Und für genau solche Probleme gibt es hier einige Lösungen.

Was die LEDs bedeuten

Die einzigen Hinweise, wie weit der Bootvorgang vorangeschritten ist, erhalten Sie von einer Reihe LEDs neben dem Soundausgang und dem USB-Port. Während das Modell B über fünf LEDs verfügt, hat das Modell A nur zwei.

- **LED 1:** Grün, „ACT“: SD-Kartenzugriff
- **LED 2:** Rot, „PWR“: 3,3-Volt-Stromversorgung vorhanden
- **LED 3:** Grün, „FDX“: LAN-Kabel angeschlossen
- **LED 4:** Grün, „LNK“: Netzwerkaktivität
- **LED 5:** Gelb, „100“: Mit 100-Mbit-Netzwerk verbunden

Da das Modell A über keine Netzwerkfunktionen verfügt, sind die letzten drei LEDs auf dessen Platine nicht vorhanden. Auch ist es möglich, dass bei älteren Modellen der B-Reihe die Bezeichnungen der LEDs ein wenig anders lauten. Die Funktionen, die sie beschreiben, bleiben trotzdem gleich.

Wenn Sie Ihr Pi an eine Stromquelle anschließen, sollte die zweite LED rot aufleuchten. Dies zeigt Ihnen an, dass Ihr Gerät mit der richtigen Menge Strom versorgt wird. Selbst wenn Sie keine Netzwerkverbindung haben oder die SD-Karte nicht angeschlossen ist, sollte die LED während des gesamten Betriebs leuchten. Im Falle eines Flackerns oder sogar eines Verlöschens haben Sie ein Problem mit der Stromversorgung Ihres Geräts. Überprüfen Sie in diesem Fall sowohl Stromkabel als auch Netzteil. Ist Ihre SD-Karte angeschlossen, so sollte die äußerste LED auf der linken Seite aufleuchten. Mit ihr wird die Zugriffsaktivität auf der SD-Karte signalisiert.

Startvorgang

Beim Start Ihres Geräts wird diese LED kurz aufleuchten, verlöschen und dann wieder flackern, während das Pi den Boot-Code von der SD-Karte einliest. Passiert dies nicht, so ist es sehr wahrscheinlich, dass entweder Ihr Boot-Code nicht richtig auf die Speicherkarte geschrieben wurde oder aber die Speicherkarte nicht mit Ihrem Pi funktioniert. Überprüfen Sie, ob Ihre SD-Karte richtig angeschlossen ist und ob jeder SD-Mikroadapter, den Sie benutzen, richtig mit der Karte verbunden ist. Prüfen Sie auch, ob die Anschlüsse zwischen SD-Kartenadapter und Konnektor des Pis identisch aussehen, da es hier in seltenen Fällen zu Kompatibilitätsproblem kommt.

Weiterhin ist es auch möglich festzustellen, an welcher Stelle des Bootvorgangs das Raspberry Pi abbricht. Im Anschluss finden Sie eine Übersicht der verschiedenen Blinkmuster der „ACT“/„OK“-LED und deren Bedeutungen. Beachten Sie jedoch, dass diese Liste eine Firmware von mindestens Mitte 2012 voraussetzt und wir diese Informationen den Raspberry Pi-Foren entnommen haben.

- **3-maliges Aufleuchten: loader.bin** nicht gefunden
- **4-maliges Aufleuchten: loader.bin** nicht geladen
- **5-maliges Aufleuchten: start.elf** nicht gefunden
- **6-maliges Aufleuchten: start.elf** nicht geladen

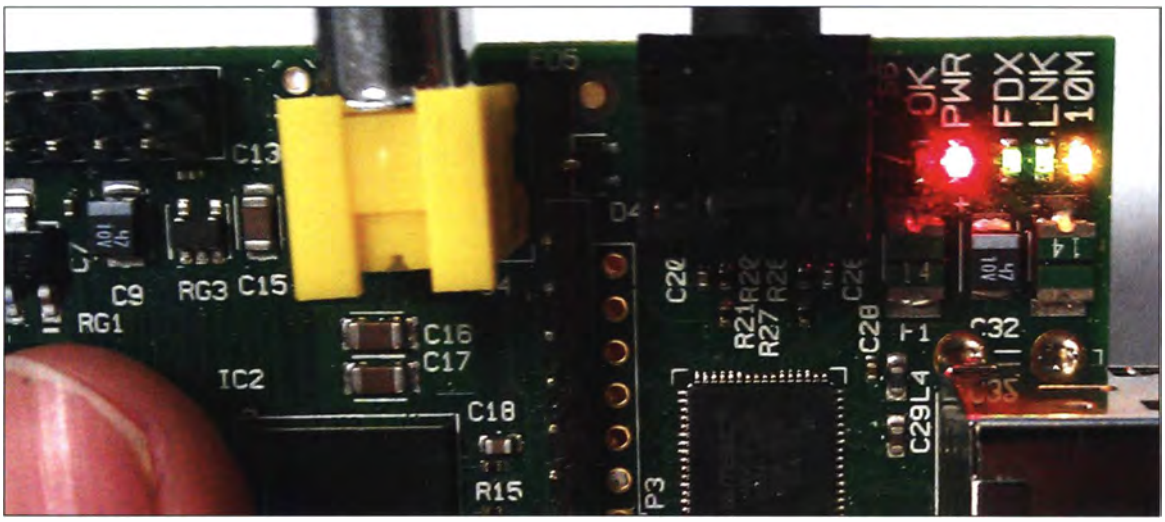
Im Rahmen eines Firmware-Updates vom Oktober 2012 wurden die Signale wie folgt neu zugeordnet:

- **3-maliges Aufleuchten: start.elf** nicht gefunden
- **4-maliges Aufleuchten: start.elf** nicht gestartet
- **7-maliges Aufleuchten: kernel.img** nicht gefunden

Es ist möglich, diese Probleme dadurch zu beheben, dass Sie auf ihrer Speicherkarte nach den oben genannten Dateien schauen und sie auf ihre Richtigkeit überprüfen. Suchen Sie nach etwas mit dem Namen Checksum und überzeugen Sie sich davon, dass die Dateien auf der SD-Karte den gleichen Checksum-Wert haben wie die Originaldateien.

Da aber solche Zugriffsfehler eher ein Hinweis darauf sind, dass entweder die SD-Karte nicht richtig ausgelesen werden kann oder aber das Betriebssystem Ihres Raspberry Pi nicht richtig auf diese geschrieben wurde, empfehlen wir Ihnen, dass Sie mit einer anderen Karte und einer anderen Methode der Image-Erstellung – unsere Installationsanleitung beschreibt Ihnen hierfür drei verschiedene Betriebssysteme – einen Versuch wagen. Während unserer Installation stießen wir jedoch

Booten



› Die hellen LEDs des Raspberry Pi sehen nicht nur schön aus, sondern können auch bei der Fehlerbehebung wichtige Hinweise liefern. Die Abbildungen zeigen eine ältere Version der Platine, auf der noch die Bezeichnungen „OK“ statt „ACT“ und „10M“ statt „100“ neben den entsprechenden LEDs aufgedruckt sind.

mit verschiedenen Kartenlesern auf Probleme. In vielen Fällen handelte es sich hierbei um solche, die in Laptops oder Notebooks verbaut waren. Daher empfehlen wir Ihnen, auf ein Einzelgerät umzusteigen. Sollten Sie Ihr Pi an einen Bildschirm angeschlossen haben, zeigen Ihnen aktuelle Versionen der Firmware einen kaleidoskopartigen Startbildschirm. Sollte der Startvorgang über diesen Bildschirm hinausgehen, aber nichts weiter passieren, liegt das Problem wiederum bei Ihrer Stromversorgung.

Netzwerk

Falls sich Ihr Raspberry Pi erfolgreich durch den initialen Startvorgang gearbeitet hat, wird nun Ihr Betriebssystem von der SD-Karte geladen. Hierbei sollte wegen des umfangreichen Datentransfers die erste LED die meiste Zeit leuchten. Die Geschwindigkeit des Bootvorgangs hängt zum größten Teil von der Geschwindigkeit der SD-Karte ab. Kurz nachdem Linux beginnt hochzufahren, blinkt zunächst die vierte LED („LNK“) und leuchten etwa eine halbe Sekunde später auch die anderen Netzwerk-LEDs. Dies zeigt an, dass die Netzwerkfunktionen aktiviert sind und sich das Gerät an Ihrem Netzwerk anmeldet. Die Netzwerk-LEDs sind vergleichbar mit denen auf der Netzwerkkarte Ihres PCs und weisen somit eher auf Probleme Ihres LAN-Netzwerks hin als auf Probleme mit der Konfiguration ihres Raspberry Pi.

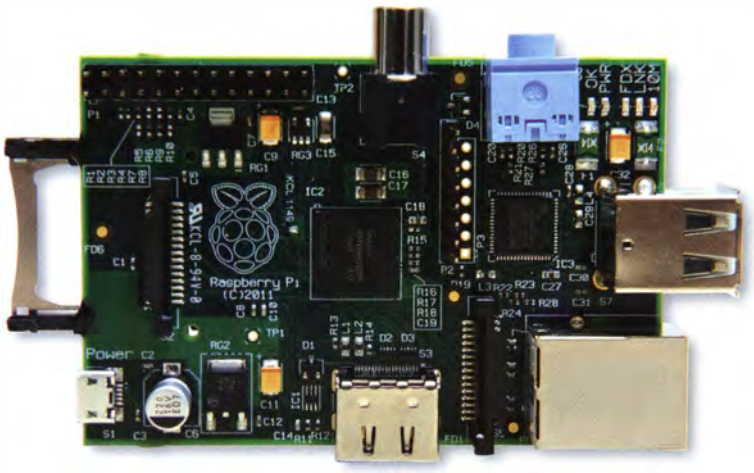
Die orangene LED signalisiert eine Full-Duplex-Verbindung. Ob es sich bei Ihrer Netzwerkverbindung um eine Full- oder Half-Duplex-Verbindung handelt, wird in dem Zusammenhang wichtig, ob Ihr Raspberry Pi gleichzeitig senden und empfangen kann. Auch wenn dies heutzutage in modernen Netzwerken nur noch selten ein Problem darstellt, hat dies natürlich Auswirkungen auf die Geschwindigkeit Ihres Geräts im Netzwerkverbund.

Die „LNK“-LED ist in Bezug auf das Netzwerk das Äquivalent zu Ihrer „ACT“-LED für den SD-Karten-Zugriff, da diese durch Blinken den Transfer von Daten im Netzwerk signalisiert. Sollte diese LED aufleuchten, können Sie davon ausgehen, dass Ihr Raspberry Pi in der Lage war, eine Verbindung zwischen Ihrem Router und sich selber herzustellen. Allerdings nur mit einer geringen Bandbreite, daher muss Ihre Netzwerkverbindung noch keine ausreichende Funktionalität für das Surfen im Internet und das Abrufen von E-Mails haben. Es ist durchaus möglich, dass trotz offensichtlicher Aktivität Ihres Raspberry Pi das Gerät selbst kein aktiver Bestandteil des Netzwerks ist. Sollten Sie trotz blinkender LED keine wirklich funktionierende Verbindung zu Ihrem Netzwerk aufbauen können, sollten Sie sich mit der Konfiguration Ihres Netzwerks befassen. Überprüfen Sie zuerst, ob Ihr Pi eine IP-Adresse von ihrem Router zugeteilt bekommen hat und wiederholen sie den Vorgang im Betriebssystem Ihres Raspberry Pi. In vielen Fällen kann das Betriebssystem die Konfiguration selber vornehmen. In manchen komplexeren Netzwerken, mit verschiedenen Domains oder mehreren Routern, können aber durchaus Probleme auftauchen. Versuchen Sie in diesem Fall zuerst, die direkteste und einfachste Verbindung zu Ihrem Netzwerk aufzubauen, und dann, sich in die komplexeren Bereiche vorzuarbeiten, bis Sie die gewünschte Konfiguration gefunden haben.

Die letzte LED zeigt Ihnen die Geschwindigkeit Ihres Netzwerks an. Sollte sie aufleuchten, erreicht Ihr Pi eine Geschwindigkeit von 100 Mbps. Anders ausgedrückt sind das 100.000.000 Datenbits pro Sekunde. Dies ist natürlich bereits ein gewaltiger Unterschied zu den 10 Mbps, mit denen Ihr Pi läuft, wenn diese LED nicht leuchtet, doch arbeiten moderne Netzwerke mittlerweile mit bis zu 1.000 Mbps. Der Grund, warum Ihr Raspberry Pi nicht in der Lage ist, eine solche Gigabitverbindung aufzubauen, ist die Tatsache, dass Ihr Ethernetadapter über einen USB2-Port implementiert wird, welcher die Transferrate auf 100 Mbps begrenzt. Dennoch reicht diese Verbindungsgeschwindigkeit für die gängigsten Anwendungen vollkommen aus. Ihr Raspberry Pi wird allerhöchstens an seine Grenzen im Netzwerkbereich stoßen, wenn Sie ihn als Hochleistungs-NAS oder zum gleichzeitigen Streamen mehrerer HD-Filme benutzen wollen. Andernfalls wird Ihnen der Unterschied in der Netzwerkgeschwindigkeit nicht auffallen und Sie können Ihr Raspberry Pi einfach genießen. ■

Befehlszeile: Erste

Lernen Sie die Befehlszeile Ihres Raspberry Pi kennen und entfesseln Sie seine volle Funktionalität, ohne die Maus zu benutzen.



Wie Sie sicherlich schon festgestellt haben, hat das Pi-Betriebssystem Raspbian eine mit Windows oder Mac OS X vergleichbare Benutzeroberfläche. Die meisten alltäglichen Aufgaben kann man innerhalb dieser Benutzeroberfläche erledigen. Es gibt einen Dateimanager, Webbrowser, Texteditor und viele weitere nützliche Anwendungen. Manchmal benötigt man allerdings eine leistungsfähigere Zugriffsmöglichkeit, und hier kommt die Befehlszeile – auch Kommandozeile, Terminal oder Shell genannt – ins Spiel.

Die Befehlszeile ist eine komplett textbasierte Benutzerschnittstelle, in die man Kommandos eingibt und daraufhin eine Ausgabe erhält. Das mag für den Neuling zunächst etwas verwirrend aussehen, doch so schlimm ist es gar nicht. Es lohnt sich, ein wenig Zeit in das Erlernen dieses Steuerungsprinzips zu investieren, denn dies wird sich später bezahlt machen.

Als Erstes muss man ein Terminal öffnen. Hierzu klickt man auf LXTerminal auf dem Raspbian Desktop.

Die folgende Zeile sollte jetzt erscheinen:

```
pi@raspberrypi ~ $
```

Das ist die Eingabeaufforderung. Wenn sie erscheint, heißt das so viel wie: Das System ist startklar für Eingaben.

Geben Sie das Kommando **pwd** ein, und drücken Sie die Enter-Taste. Folgendes sollte zu sehen sein:

```
/home/pi
```

Falls man seinen Benutzernamen geändert hat, sieht man eine andere Zeile. Der eher kryptisch benannte **pwd**-Befehl steht für „print working directory“ und gibt ganz einfach das Verzeichnis aus, in dem man sich gerade befindet. Wenn Sie das Terminal öffnen, startet es in Ihrem Home-Verzeichnis (mehr dazu auf Seite 32 in diesem Heft).

Nun wissen wir also, wo wir uns befinden. Der nächste logische Schritt ist, sich durch die Verzeichnisstruktur zu bewegen. Hierzu verwendet man das Kommando **cd** („change directory“). Geben Sie Folgendes ein:

```
cd ..
pwd
```

Das System sollte zu **/home** zurückkehren. Das kommt daher, dass wir in das „ ..“-Verzeichnis gewechselt haben. Zwei Punkte verweisen immer auf das übergeordnete Verzeichnis. Um zum Home-Verzeichnis zurückzukehren, kann man **cd pi** eingeben. Hierfür gibt es auch noch einen anderen Weg. Die Tilde (das Zeichen „~“) verweist immer auf Ihr Home-Verzeichnis. Wo auch immer man sich im Verzeichnisbaum befindet, gelangt man über die Eingabe von **cd ~** dorthin zurück.

Geben Sie nun **ls** ein, und drücken Sie die Enter-Taste. Dies listet alle Dateien im aktuellen Verzeichnis auf. Einer der großen Vorteile von Befehlen ist, dass wir festlegen können, wie sie ausgeführt werden. Dies geschieht mit der Hilfe von Parametern, die man nach dem Befehl setzt und die mit einem „-“ beginnen. Wenn man zum Beispiel alle Dateien im aktuellen Verzeichnis auflisten möchte (einschließlich der versteckten, die bei unixbasierten Systemen mit „.“ beginnen), nutzen wir den Parameter **-a**. Schreiben Sie in Ihrem Terminal also **ls -a**.

Jetzt sollten mehr Dateien erscheinen. Ein anderer Parameter für **ls** lautet **-l**. Dieser gibt uns weitere Informationen zu jeder Datei. Man kann sogar Parameter kombinieren wie zum Beispiel **ls -al**.

Welche Befehle sollte man kennen?

Sie fragen sich jetzt vielleicht, woher um alles in der Welt Sie wissen sollen, welche Befehle und Parameter Sie für eine bestimmte Aufgabe verwenden können. Es gibt eine gute und eine schlechte Nachricht. Die gute ist, dass es normalerweise nicht allzu schwierig ist, die Parameter für einen Befehl herauszufinden, in aller Regel geht dies mittels **-h** oder **--help**. Wenn man beispielsweise **ls --help** ausführt, erhält man eine lange Liste von Parametern und deren Bedeutung, einschließlich der folgenden:

```
-a, --all    Auch mit Punkt beginnende Dateien anzeigen
...
-l           Alle Dateidetails anzeigen
```

Die zweite Möglichkeit, Informationen zu einem Befehl zu

Interaktive Programme

Die meisten der Befehle, die wir hier behandeln, sind nicht interaktiv. Das bedeutet, man startet sie und wartet, bis sie fertig ausgeführt sind. Es funktionieren jedoch nicht alle Befehlszeilenprogramme auf diese Weise. Als Sie beispielsweise anfangs Raspbian hochgefahren haben, wurde ein Config-Tool, das im Terminal läuft, gestartet. Es gibt noch ein paar Programme, die ähnlich funktionieren, dazu zählen insbesondere die Texteditoren. Sie ermöglichen ein Bearbeiten von Dateien ohne grafisches Benutzerinterface. Es gibt einige recht komplizierte, die schwierig zu erlernen sind, die aber wirklich interessant werden, wenn man viel von der Befehlszeile aus arbeitet. Es gibt aber auch einen einfach zu bedienenden terminalbasierten Texteditor namens *Nano*. Geben Sie im Terminal **nano** ein, gefolgt von einem Dateinamen. Dann können Sie im ausgegebenen Text navigieren und Änderungen durchführen. Speichern Sie alles mit Ctrl+X, und kehren Sie zur Eingabeaufforderung zurück.

Schritte

bekommen, ist die Verwendung von **man**. Dies ist die Abkürzung für Handbuch (engl.: manual). Man gibt **man** gefolgt von dem jeweiligen Befehlsnamen ein und erhält die Hilfe zu diesem Befehl. Um die Hilfe für **ls** zu sehen, geben Sie **man ls** ein. Zum Navigieren auf der Seite verwendet man die Aufwärts- und Abwärts-Pfeiltasten oder – zum schnelleren Scrollen – die Bild-nach-oben- und Bild-nach-unten-Tasten. Um nach einem Wort oder einer Formulierung auf der Handbuchseite zu suchen, geben Sie / und dann den jeweiligen Suchbegriff ein. Mit den Kommando /**-I** lassen Sie sich beispielsweise alle Stellen anzeigen, wo **-I** vorkommt. Mit der N-Taste und Shift+N kann man zwischen den Treffern vor und zurück navigieren.

Da wir noch weitere Befehle einführen, wäre es gut, wenn Sie sich die Hilfe und die Handbuchseite ansehen und sich mit ihnen vertraut machen. Natürlich können Sie auch immer einen Befehl bei Google nachschlagen, falls Ihnen der Umgang mit der textbasierten Hilfe nicht so sehr liegt. Allerdings werden Sie schneller mit der Befehlszeile vertraut, wenn Sie mit dem Terminal arbeiten.

Hilfefunktionen

Nun kommt aber noch die schlechte Nachricht. Es kann nämlich manchmal knifflig sein, die richtigen Befehle zu finden, wenn man nicht weiß, wie sie heißen. Dann ist die **man**-Stichwortsuche eine nützliche Anwendung. Hierzu verwendet man den Parameter **-k** („keyword").

Um zum Beispiel nach allen Programmen zu suchen, die mit dem Browser Ihres Systems zu tun haben, führen Sie **man -k browser** aus. Sie werden sehen, dass dieser Befehl sowohl Programme der grafischen Oberfläche als auch Befehlszeilenkommandos auflistet. Das liegt daran, dass zwischen den beiden kein wirklicher Unterschied besteht. Sie können Fenster vom Terminal aus starten und manchmal sogar steuern.

Falls Sie *Iceweasel* (eine spezielle Version von *Firefox*) auf Ihrem Pi installiert haben (dies gehört nicht zu den vorinstallierten Programmen), dann können Sie **TuxRadar.com** in einem neuen Reiter im geöffneten Iceweasel-Fenster mit dem Befehl **iceweasel --new-tab www.tuxradar.com** öffnen.

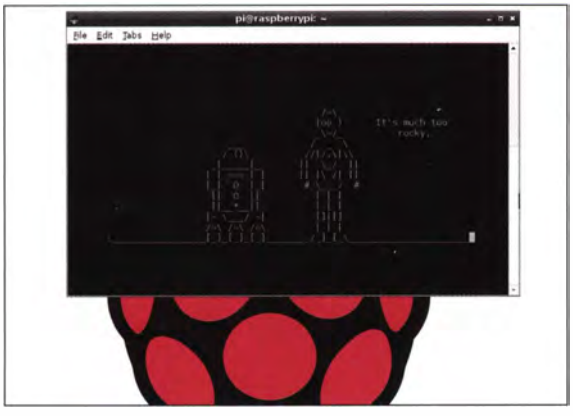
Hier noch in aller Kürze ein paar weitere nützliche Befehle: **rm** löscht eine Datei. **mkdir** erzeugt ein neues Verzeichnis. Mit **cp** kann man eine Datei an einen anderen Speicherort kopieren. Dieser Befehl benötigt zwei Parameter (für die Originaldatei und die neue Datei). **cat** gibt den Inhalt von einer oder mehreren Textdateien aus. Der Befehl nimmt beliebig viele Parameter und zeigt die Ausgabe direkt im Terminal an. **less** ist eine benutzerfreundlichere Möglichkeit, sich Texte anzusehen, man kann mit den Pfeiltasten rauf und runter scrollen. Um vom Programm zurück zur Befehlszeile zu gelangen, drücken Sie die Q-Taste. **find** ist ein nützliches Kommando, um Dateien auf Ihrem Computer zu suchen. Sie verwenden ihn im Format **find location flags**. Wenn Sie zum Beispiel alle Dateien auf Ihrem Computer finden möchten, die tags zuvor geändert wurden, führen Sie folgenden Befehl aus:

```
find / -mtime 1
```

Auf Seite 27 in diesem Heft finden Sie weitere Informationen zur Benutzung dieses und der anderen genannten Befehle.

Tabulator-Vervollständigung

Wenn man mit langen Dateinamen arbeitet, ist es anstrengend, sie jedesmal neu zu schreiben, wenn man einen Befehl mit Ihnen ausführen möchte. Um das zu erleichtern, gibt es im Terminal die Tabulator-Vervollständigung. Wenn man also mit der Eingabe eines Dateinamens beginnt und die Tabulator-Taste drückt, versucht das System, den restlichen Dateinamen zu ergänzen. Falls es nur einen Dateinamen gibt, der mit dem übereinstimmt, was bisher eingegeben wurde, wird der restliche Dateiname automatisch ergänzt (geben Sie **cd /h** ein, und drücken Sie die Tabulator-Taste). Falls es mehrere Dateinamen gibt, erfolgt die Vervollständigung nur so weit, wie diese übereinstimmen. Ein wiederholtes drücken der Tabulator-Taste zeigt die möglichen Optionen an (geben Sie **cd /m** ein, und drücken Sie die Tabulator-Taste zweimal).



› Man kann sich sogar Filme in der Befehlszeile ansehen. Um ein paar Klassiker zu streamen, geben Sie einfach telnet towel.blinkenlights.nl ein und holen Sie sich Popcorn.

Mächtigere Funktionen

Bis hierher hätte man die entsprechenden Funktionen auch mit der grafischen Benutzeroberfläche nutzen können, ohne sich Befehle und Parameter merken zu müssen, doch jetzt kommen wir zu den speziellen Stärken der Eingabe per Kommandozeile.

Wir beginnen mit den sogenannten Wildcards. Das sind Platzhalter, die für beliebige andere Zeichen stehen können. Am besten lässt sich dies an einigen Beispielen veranschaulichen.

Als Erstes erstellen wir dazu ein neues Verzeichnis und legen darin ein paar leere Dateien an (Letzteres geht ganz einfach mit dem Befehl **touch**, der eine leere Datei mit dem Namen eines jeden angehängten Parameters erzeugt).

```
mkdir wildcards
cd wildcards
touch one two three four
touch one.txt two.txt three.txt four.txt
```

»

» Führen Sie **ls** aus, und schauen Sie nach, welche Dateien sich in der neuen Verzeichnisstruktur befinden. Es sollten acht sein.

Die erste Wildcard, die wir verwenden, ist „*“. Dieser Platzhalter steht für eine beliebige Zeichenkette von null oder mehreren Zeichen. Verwendet man ihn alleine, zeigt er alle Dateien in der Verzeichnisstruktur auf. Probieren Sie es aus:

```
ls *
```

Das ist noch nicht besonders nützlich, doch wir können noch weitere Zeichen hinzunehmen. Was denken Sie? Wird ***.txt** funktionieren? Testen Sie, ob Sie richtig liegen:

```
ls *.txt
```

Was ist mit **one***? Probieren Sie es auch hiermit.

```
ls one*
```

Die Wildcards können mit allen Befehlen der Kommandozeile verwendet werden. Sie sind besonders nützlich zum Sortieren von Dateien. Um alle .txt-Dateien in ein neues Verzeichnis zu kopieren, könnten Sie folgende Befehle ausführen:

```
mkdir text-files
cp *.txt text-files
```

Ob alles geklappt hat, können wir folgendermaßen überprüfen:

```
ls text-files/
```

Der zweite Platzhalter, den wir kennenlernen ist **„?“**. Er steht für ein beliebiges einzelnes Zeichen. Was ergibt wohl:

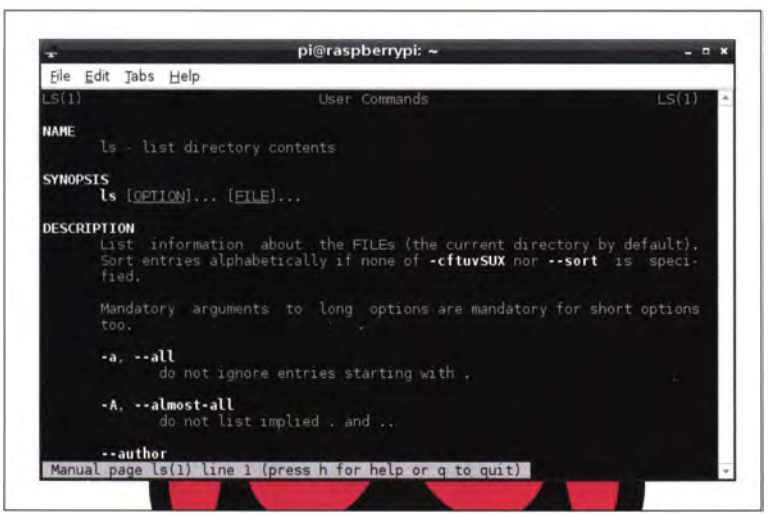
```
ls ???
```

Raten Sie, und probieren Sie es einfach aus.

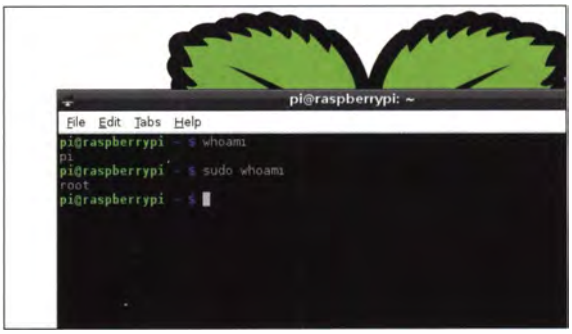
Wir können auch Platzhalter für bestimmte Zeichen festlegen. **[abc]** steht für ein kleingeschriebenes A, B oder C. Was, denken Sie, zeigt uns der Befehl **ls [ot]*** an? Versuchen Sie es!

```
ls [!ot]*
```

Welchen Unterschied hat das Ausrufezeichen ausgemacht? Alles, was nicht mit einem kleingeschriebenem O oder T beginnt, sollte aufgelistet werden.



› man ist wahrscheinlich der wichtigste Befehl in jedem unixähnlichen System. Wenn man ihn versteht, versteht man auch jeden anderen Befehl. Nehmen Sie sich Zeit, um mit seinem Aufbau und seiner Sprache vertraut zu werden. Das wird Ihnen das Leben leichter machen.



› Verwenden Sie den sudo-Befehl, um zwischen dem normalen Benutzer „pi“ und dem Super-Anwender „root“ zu wechseln.

Ausgabeumlenkung

Die Befehle, die wir bisher verwendet haben, haben die entsprechenden Ergebnisse im Terminal angezeigt. Meistens ist das auch genau das, was wir beabsichtigen. Doch manchmal ist eine andere Ausgabeform hilfreicher. Unter Linux hat man noch zwei weitere Ausgabemöglichkeiten: in eine andere Datei oder ein anderes Programm.

Um die Ausgabe in eine andere Datei umzulenken, verwendet man das Zeichen „>“, gefolgt von dem Dateinamen. Führen Sie folgende Befehle aus:

```
ls > files
cat files
```

Sie werden sehen, dass eine neue Datei namens **files** erzeugt worden ist, die die Ergebnisse von **ls** enthält.

Die Möglichkeit, die Ergebnisse an ein anderes Programm zu senden, ist eine weitere, wirklich nützliche Anwendung der Linux-Befehlszeile, da man hier eine Reihe von Befehlen zusammenfassen kann. Es gibt Befehle, die darauf ausgelegt sind, auf diese Art verkettet zu werden. Hier ein einfaches Beispiel dafür: Wenn Sie **find** / ausführen würden, würde Ihnen jede Datei Ihres Systems aufgelistet. Dies wäre eine sehr lange Liste von Dateinamen, die nicht auf eine Bildschirmseite passt. Statt uns das Ergebnis direkt auf dem Bildschirm ausgeben zu lassen, können wir jedoch die Liste an einen anderen Befehl senden, der sie lesefreundlicher darstellt. Hierzu können wir den **less**-Befehl verwenden, den wir weiter oben bereits kennengelernt haben. Führen Sie Folgendes aus:

```
find / | less
```

Weiterführende Quellen

Sie kennen nun die Grundlagen der Eingabe per Befehlszeile. Dieses Rüstzeug sollte Ihnen für den Anfang genügen, und wir hoffen, dass Sie mit der Zeit die vielfältigen Möglichkeiten des Terminals für sich entdecken.

Falls Sie mehr über die Benutzung der Befehlszeile erfahren möchten (was wir natürlich sehr empfehlen), gibt es hierzu eine Vielzahl von Quellen sowohl in gedruckter Form als auch online. Eine gute Einstiegsmöglichkeit bietet **www.linuxcommand.org**. Dort ist auch ein Link zu einer kostenlosen PDF-Version des Buches *The Linux Command Line* zu finden. ■

sudo

Für die meisten Anwendungsfälle ist es für Rasperry-Pi-Nutzer ausreichend, im Home-Verzeichnis zu arbeiten (z.B. **/home/pi**). Man kann auch die meisten anderen Dateien ansehen, aber nicht ändern. Das Installieren von Software ist ebenfalls nicht möglich. Linux beinhaltet nämlich ein Berechtigungssystem, welches gewöhnliche Nutzer davor bewahrt, versehentlich Systemeinstellungen zu ändern. Möchte man jedoch Systemeinstellungen ändern, kann man **/sudo** nutzen und einen Befehl als Super-User (manchmal auch als „root“ bezeichnet) ausführen. Ein Super-User kann so ziemlich alles ausführen. Sie müssen einfach nur **sudo** vor den Befehl setzten. Zum Beispiel:

```
sudo apt-get install synaptic
```

Mit diesem Befehl wird das Paket synaptic installiert und für alle Benutzer zugänglich gemacht.

Die wichtigsten Befehle im Überblick

Navigation und Dateien

» **df -h** zeigt die freie Speicherkapazität des Gerätes an.
» **pwd** zeigt das Verzeichnis an, in dem man sich gerade befindet.
» **cd** Verzeichniswechsel. **cd movies** navigiert zum Verzeichnis **movies**, mit **cd ~** gelangen Sie in Ihr Home-Verzeichnis, mit **cd /** zum Hauptverzeichnis und mit **cd ..** in die nächsthöhere Verzeichnisebene.
» **ls** Dateiliste für das aktuelle oder per Parameter angegebene Verzeichnis. **ls movies** listet die Dateien des Verzeichnisses **movies** auf, **ls -a** zeigt alle Dateien (auch die versteckten) an, und **ls -l** zeigt zusätzliche Informationen zu jeder Datei an.
» **cp** Kopieren. **cp orig-file new-file** kopiert **orig-file** zu **new-file**.
» **wget** Download von Dateien aus dem Internet. Um die Google-Homepage herunterzuladen, verwendet man **wget www.google.com**

Dateien suchen

» **find <location> <tests>**. Nützliche Parameter sind:
» **-mtime <Anzahl>** zeigt alle Dateien an, die in den letzten **<Anzahl>** Tagen geändert wurden. Anstelle von **<Anzahl>** kann beispielsweise 2 (vor genau 2 Tagen), -2 (vor weniger als zwei Tagen) oder +2 (vor mehr als zwei Tagen) stehen.
» **-name <Dateiname>** sucht Dateien mit dem Namen **<Dateiname>**.
» **-iname <Dateiname>** findet Dateien namens **<Dateiname>** ohne Beachtung der Groß- und Kleinschreibung.
» **-writable** sucht alle Dateien, die beschreibbar sind.
» Es können auch Parameter kombiniert werden. Zum Beispiel listet **find / -mtime -2 -writeable** alle Dateien im Dateisystem auf, die vor weniger als 2 Tagen geändert wurden und für den aktuellen Benutzer beschreibbar sind.

Fernzugriff

» **ssh** Loggen Sie sich auf einem entfernten Rechner über Secure Shell (SSH-Protokoll) ein. **ssh pi@192.168.1.2** wird sich als Anwender „pi" auf dem Computer mit der IP-Adresse **192.168.1.2** einloggen. Beachten Sie, dass dies nur funktioniert, wenn auf dem Remote-Rechner ein SSH-Server läuft.
» **scp** Secure Copy. **scp <Datei> pi@192.168.1.2:/home/pi** kopiert **<Datei>** in das Verzeichnis **/home/pi** auf dem Rechner **192.168.1.2**, und **scp pi@192.168.1.2:/home/pi/<Datei>** kopiert **/home/pi/<Datei>** von dem Rechner **192.168.1.2** in das aktuelle Verzeichnis. Achtung, dies ist nur möglich, wenn auf dem Remote-Rechner ein SCP-Server läuft.

Wildcards

» ***** ist Platzhalter für eine beliebige Zeichenkette (auch leere).
» **?** ist Platzhalter für ein einzelnes Zeichen.
» **[abc]** ist Platzhalter für a, b oder c.
» **[!abc]** ist Platzhalter für alle Zeichen außer a, b oder c.
» **[A-Z]** ist Platzhalter für einen Buchstaben von A-Z (großgeschrieben).
» **[A-z]** ist Platzhalter für einen Buchstaben von A-z (klein- oder großgeschrieben).
» **[eins, zwei]** ist Platzhalter für die Wörter eins und zwei.

Informationen über den Computer

» **top** zeigt das Programm an, das gerade die meiste CPU- und Speicherkapazität benötigt.
» **uname** zeigt Informationen zum Kernel an. **uname -m** zeigt Informationen zur Hardware an.
» **lscpu** gibt Informationen über die CPU aus.
» **dmesg** gibt Kernel-Meldungen über den Bildschirm aus (und kann hilfreich sein, um Hardwareprobleme zu identifizieren).

Textdateien

» **head** gibt die ersten 10 Zeilen einer Textdatei auf dem Bildschirm aus. Mit dem Parameter **-n** kann man die Anzahl 10 ändern Beispielsweise zeigt **dmesg | head -n 15** die ersten 15 Zeilen einer Kernel-Meldung an.
» **tail** zeigt die letzten 10 Zeilen einer Textdatei auf dem Bildschirm an. Der Parameter **-n** kann wie gehabt verwendet werden. Man kann auch Dateiänderungen mit dem Parameter **-f** („follow") verfolgen. So zeigt **tail -n 15 -f /var/log/syslog** die letzten 15 Zeilen der Log-Datei des Systems an und aktualisiert laufend die Ausgabe bei Änderungen.
» **less** ermöglicht das Scrollen innerhalb einer Textdatei.
» **cat** gibt den Inhalt einer Textdatei im Terminal aus.
» **nano** Das Programm *Nano* ist ein benutzerfreundlicher Befehlszeilen-Texteditor (Ctrl+X schließt das Programm und fragt ab, ob Änderungen gespeichert werden sollen).

Besondere Tastenkombinationen

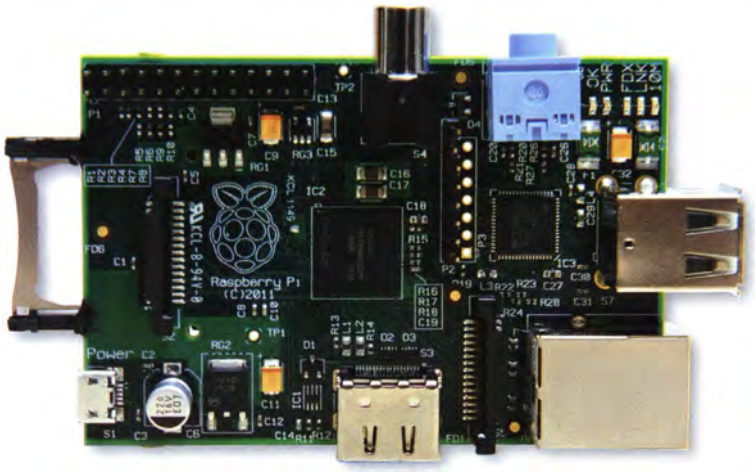
» **Ctrl+C** beendet jegliches Programm, das gerade im Terminal läuft.
» **Ctrl+D** sendet das Dateiendezeichen an jegliches Programm, das gerade im Terminal läuft.
» **Ctrl+Shift+C** kopiert ausgewählten Text in die Zwischenablage.
» **Ctrl+Shift+V** fügt den Text aus der Zwischenablage ein.

Software installieren

» **./configure** Wenn man den Quellcode eines Programms entpackt, wird gewöhnlich ein neues Verzeichnis, das das Programm beinhaltet, erstellt. Wechseln Sie in dieses Verzeichnis und führen Sie **./configure** aus, um zu kontrollieren, ob Ihr System über alles Nötige zum Kompilieren der Software verfügt.
» **make** kompiliert die Software.
» **make install** (benötigt **sudo**) verschiebt die gerade kompilierte Software an einen geeigneten Ort im System, sodass man sie wie einen normalen Befehl ausführen kann.
» **apt-get** kann dazu verwendet werden, Software zu installieren oder zu deinstallieren. So wird per **sudo apt-get install iceweasel** das Programm *Iceweasel* (eine Variante von *Firefox*) installiert und per **sudo apt-get purge iceweasel** wieder entfernt.
Mit **apt-get update** rufen Sie eine aktuelle Liste aller Datenpakete von der Repository-Datenbank ab (eine gute Idee, bevor Sie irgendetwas unternehmen), und **apt-get upgrade** führt ein Upgrade aller Datenpakete aus, die in der Repository-Datenbank eine neuere Version beinhalten.
» **apt-cache search <keyword>** sucht in der Repository-Datenbank nach allen Paketen, die ein bestimmtes Stichwort enthalten.

Einstellungssache:

Mithilfe des Konfigurationswerkzeugs raspi-config können Sie einige grundlegende Einstellungen Ihres Raspberry Pi vornehmen.



Sie haben nun die notwendige Hardware samt Zubehör beisammen und auch das Betriebssystem installiert – jetzt kann es also losgehen mit dem Raspberry Pi. Allerdings lohnt es sich, vor dem Herumexperimentieren noch einige weitere Vorbereitungen zu treffen. Sie können Ihren Pi nämlich in hohem Maße ihren persönlichen Bedürfnissen und Vorlieben anpassen.

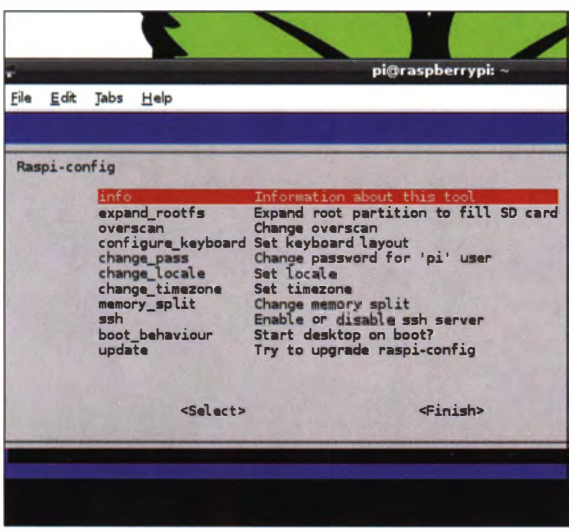
Diese Anpassungen lassen sich mit der Software raspi-config vornehmen, die im Lieferumfang der Raspbian-Distribution bereits enthalten ist. Sie können das Programm also sofort aufrufen, ohne noch etwas installieren zu müssen.

Beim ersten Booten des Betriebssystems lädt raspi-config automatisch, aber falls Sie es dabei übersprungen haben oder später erneut etwas ändern möchten, rufen Sie es ganz einfach wieder auf, indem Sie die Kommandozeile öffnen und den Befehl **sudo raspi-config** eingeben. Daraufhin erscheint eine Passwortabfrage, die Sie mit „raspberry“ beantworten – dies ist das voreingestellte Standardpasswort.

Um ein Fenster mit der Kommandozeile zu öffnen, können Sie entweder auf das Icon LXTerminal auf dem Raspbian-Desktop klicken oder die Tastenkombination STRG+ALT+T drücken.

Eine der tollen Eigenschaften des Raspberry Pi ist es, dass Sie die Grundeinstellungen des Konfigurationstools immer unverändert lassen können, wenn Sie unsicher sind, welche Option Sie wählen sollen – der Pi wird trotzdem problemlos funktionieren. Und falls Sie etwas an den Einstellungen ändern und dabei einen Fehler machen, dann können Sie jederzeit zurück zu raspi-config gehen und das Problem beheben.

Schritt für Schritt: Konfiguration



1 Menüliste

Der Menübildschirm zeigt an, welche Eigenschaften konfiguriert werden können. Im Folgenden gehen wir die Optionen Punkt für Punkt durch. Hier im Bild fehlt die Option **overclock** (zwischen **memory_split** und **ssh**), die im Zuge einer Überarbeitung von *raspi-config* Ende 2012 ergänzt wurde. Infos zum Overclocking des Pi finden Sie auf Seite 66.

2 Dateisystem vergrößern

Sofern Sie bei der Partitionierung dem Standardschema gefolgt sind, belegt Raspbian etwa 1,6 GB auf der SD-Karte. Daher empfiehlt es sich, einen Speicher mit einem Volumen von mindestens 2 GB zu verwenden. Falls Sie eine größere Karte haben, erweitern Sie das Dateisystem mithilfe dieser Option so, dass Sie den restlichen Platz ebenfalls nutzen können.

Konfiguration

3 Overscan

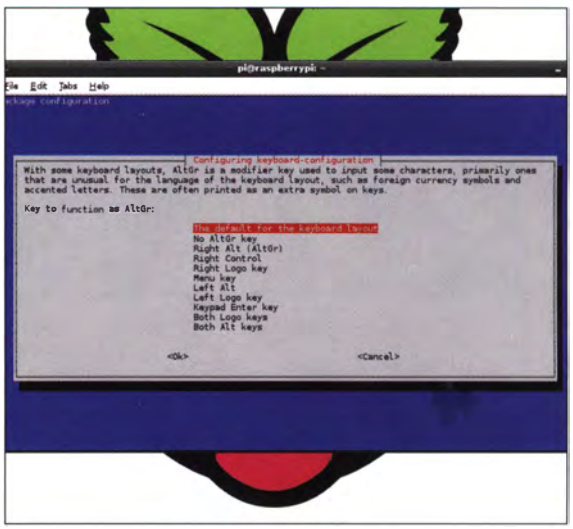
Sie können einen Fernseher oder einen Computerbildschirm als Ausgabegerät für Ihren Raspberry Pi verwenden. Je nach Gerät kann es vorkommen, dass ein breiter schwarzer Rand um das angezeigte Bild herum besteht. Mithilfe des Overscan-Umschalters können Sie dies beheben.

4 Tastatur erkennen

Da der Raspberry Pi ohne Keyboard ausgeliefert wird, muss er in der Lage sein, unterschiedliche angeschlossene Tastaturen korrekt zu erkennen. Bei den Konfigurationsoptionen findet sich eine Liste vieler Keyboardmodelle unterschiedlicher Hersteller, aus der Sie das Ihre auswählen können.

5 Tastaturlayout

Mit dieser Option können Sie die Tastaturbelegung einstellen. In englischsprachigen Ländern werden meist QWERTY-Belegungen verwendet, in Deutschland und Österreich das QWERTZ-Layout. Die Schweiz hat eine eigene Belegung. Da es den Buchstaben ß in der Schweizer Orthographie des Deutschen nicht gibt, fehlt er dort auch auf der Tastatur.



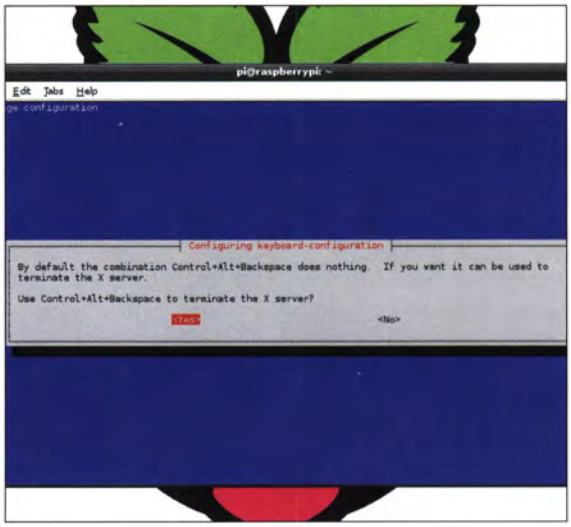
6 Sonderzeichentaste

Beim Raspberry Pi können Sie einstellen, welche Taste des Keyboards als Sonderzeichentaste fungieren soll. Standardmäßig ist dies die ALT GR-Taste. Damit erzeugen Sie Sonderzeichen wie etwa das Euro-Symbol € oder das @.

»

7 Compose-Taste

Auch als Compose-Taste können Sie eine bestimmte Taste definieren. Wenn Sie die Compose-Taste drücken, werden die nächsten Tastentipper so behandelt, als wären sie gleichzeitig erfolgt. Dies erleichtert die Eingabe von Tastenkombinationen wie zum Beispiel STRG+ALT+T.



8 STRG+ALT+Backspace

Wenn auf einem Linux-System die grafische Benutzeroberfläche einfriert, kann man mittels der Tastenkombination STRG+ALT+Backspace den Grafikserver abschalten und in einen Nur-Text-Modus wechseln. Standardmäßig ist diese Funktion bei Rasbian deaktiviert, aber Sie können sie hier im Konfigurationsmenü aktivieren.

9 Passwort ändern

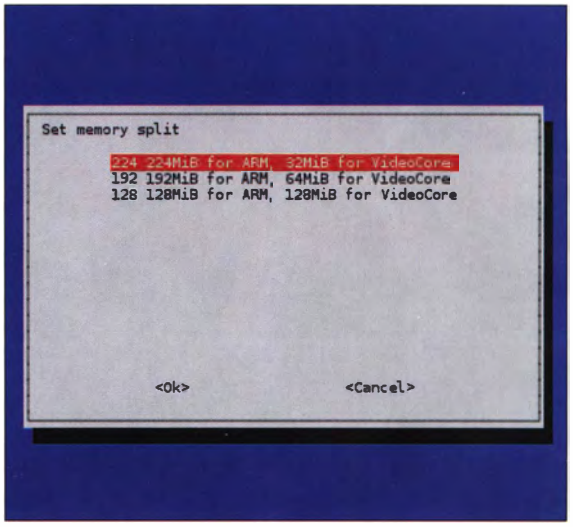
In der unveränderten Grundeinstellung heißt der Standardbenutzer „pi“ und ist diesem das Passwort „raspberry“ zugewiesen. Sie sollten das Passwort unbedingt ändern, um Ihren Rechner vor unbefugten Zugriffen zu schützen. Wählen Sie ein ausreichend sicheres (also langes) Passwort.

10 Gebietseinstellungen

Das Gebietsschema oder Locale umfasst regions- oder länderspezifische Einstellungen zur Systemsprache, zum Zeichensatz, zum Zeitformat und zu anderen Dingen. Wenn Sie die Standardeinstellungen für Deutschland haben möchten, wählen Sie „de_DE.UTF-8 UTF-8“ aus.

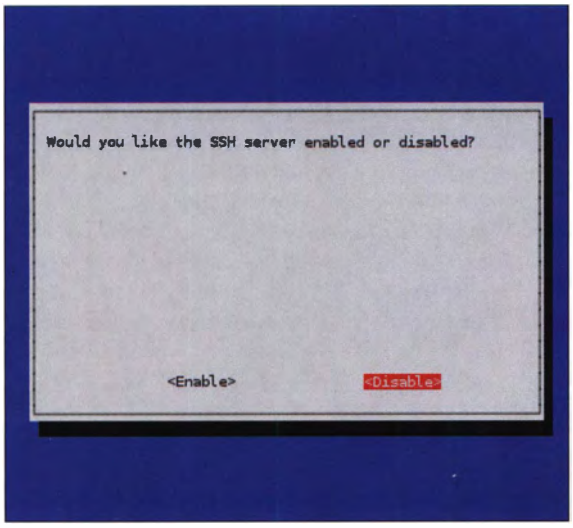
11 Zeitzone

Damit die Systemzeit Ihres Raspberry Pi der Ortszeit Ihres Standorts entspricht und Dateien korrekte Speicherdaten erhalten, müssen Sie Ihre Zeitzone einstellen. Die Zuordnung erfolgt via Kontinent und Hauptstadt/Metropole der einzelnen Staaten.



12 Speicheraufteilung

Der Raspberry Pi besitzt einen Hauptspeicher von 512 MB RAM (Modell B) beziehungsweise 256 MB RAM (Modell A), der zwischen der CPU und dem Grafikprozessor aufgeteilt ist. Falls Sie grafikintensive Programme verwenden, können Sie den Anteil des Grafikprozessors erhöhen. Standardmäßig bekommt er eine Quote von 64 MB RAM, diese können Sie auf 128 MB verdoppeln. Sie können den Anteil des Grafikchips aber auch verkleinern.



13 SSH aktivieren

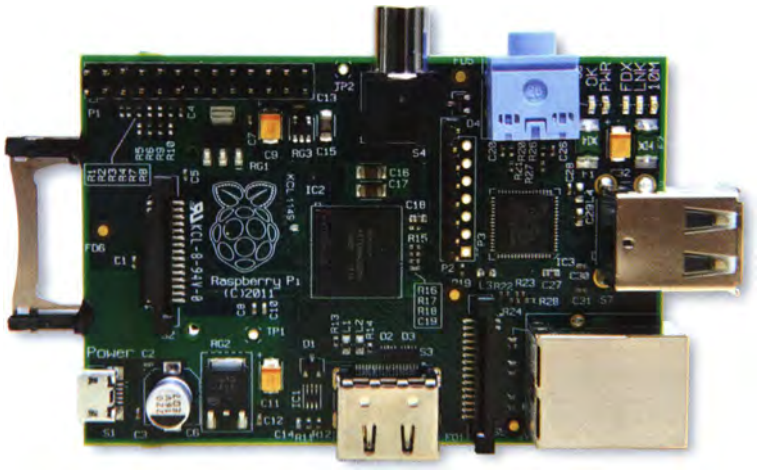
Das Netzwerkprotokoll SSH erlaubt Ihnen die Einrichtung einer geschützten Verbindung zwischen dem Raspberry Pi und einem anderen Computer (siehe auch Seite 52-55). Falls Sie das Protokoll jedoch nicht verwenden, schalten Sie es ruhig ab, da es immer sicherer ist, möglichst wenig Internetdienste im Hintergrund laufen zu haben.

14 Start mit grafischer Benutzeroberfläche

Falls Sie beim Hochfahren Ihres Raspberry Pi ohne Umschweife die grafische Benutzeroberfläche angezeigt bekommen möchten, wählen Sie hier die Option „Yes“. Andernfalls wird zunächst in die textbasierte Konsole gebootet, und Sie müssen das grafische Interface manuell mittels des Befehls **startx** aufrufen. ■

Das Dateisystem:

Lernen Sie, wie Linux Dateien verwaltet und wie Sie zusätzliche Speichergeräte zu Ihrem Raspberry Pi hinzufügen.



Das Linux-Dateisystem ist unter dem Wurzelverzeichnis (engl. root directory) aufgehängt, welches durch einen Schrägstrich (/) symbolisiert wird. Wenn man das Dateisystem mit einem Haus vergleicht, dann entspricht / der Haustüre, durch die man ins Haus gelangt. Wir probieren das mal aus. Hierzu öffnen wir ein Terminal und geben Folgendes ein:

```
cd /
ls
```

Uns werden daraufhin einige Dateien und Verzeichnisse ausgegeben. In unserer Metapher entsprechen die Verzeichnisse Türen zu weiteren Räumen. Der wichtigste Raum ist für die meisten Anwender ihr Home-Verzeichnis, das wir uns ansehen können, wenn wir folgende Befehle eingeben:

```
cd /home
ls
```

Haben Sie den / vor **home** bemerkt? Dieser bedeutet, dass home sich im Wurzelverzeichnis (/) befindet. Dateipfade, die mit / beginnen, werden als absolute Pfade bezeichnet. Über den Befehl **ls** werden uns dann alle Dateien in **/home** angezeigt. Für jeden angelegten Benutzer auf Ihrem Raspberry Pi sollte es hier ein eigenes Verzeichnis geben. Zum Verzeichnis des Benutzers pi gelangt man mit dem Befehl:

```
cd pi
```

Diesmal haben wir vor den Verzeichnisnamen kein / gesetzt. Das heißt, dass sich das Verzeichnis des Benutzers pi im aktuellen Verzeichnis befindet. In diesem Fall spricht man von einem relativen Pfad. Wir hätten auch den absoluten Pfad mit **cd /home/pi** aufrufen können.

Wenn wir uns das Dateisystem als Gebäudekomplex mit vielen Zimmern vorstellen, so ist es wichtig zu wissen, dass sich nicht alle Verzeichnisse auf demselben Speichergerät befinden. Wir haben es also mit Zimmern in unterschiedlichen Gebäuden zu tun, die durch Gänge miteinander verbunden sind. Für den ungeübten Benutzer ist dies nicht zu erkennen. Im Fall eben hätte sich das Benutzerverzeichnis **pi** auf einem physisch anderen Laufwerk als **home** befinden können, und wir hätten es niemals bemerkt. Wem die Verzeichnisstruktur von Windows mit den getrennten Laufwerken vertraut ist, dem mag dies seltsam vorkommen, doch es sorgt für deutlich mehr Flexibilität.

Um anzuzeigen, welche Geräte gerade an Ihr System angeschlossen sind, geben Sie Folgendes ein:

```
df -h
```

Es werden eine paar Zeilen ausgegeben, deren erste Spalte anzeigt, wo sich das Dateisystem befindet (dies ist nicht der Pfad, an dem das Speichergerät eingebunden wird – also die Tür in unserem Vergleich –, sondern die Datei für das interne Dateisystem). Es gibt zwei unterschiedliche Typen: diejenigen, die mit **/dev/** beginnen, und welche, die anders anfangen. Die Letzteren stellen keine echten Geräte dar, sondern vom Betriebssystem erzeugte virtuelle Geräte, die virtuelle Dateien mit Systeminformationen beinhalten. Schauen Sie sich zum Beispiel einmal an, was sie unter **/dev** finden: für jede Hardware Ihres Systems eine Datei, einschließlich aller Laufwerke. Daher haben die Ausgabezeilen, die sich auf physische Geräte beziehen, alle ein **/dev/** in der ersten Spalte.

Die Geräte, die mit **/dev/** beginnen, werden wiederum in zwei Kategorien eingeteilt: solche, die mit **/dev/mmcblk** beginnen (das sind die auf einer SD-Karte), und solche, die mit **/dev/sd** beginnen (das sind die auf zusätzlichen Geräten).

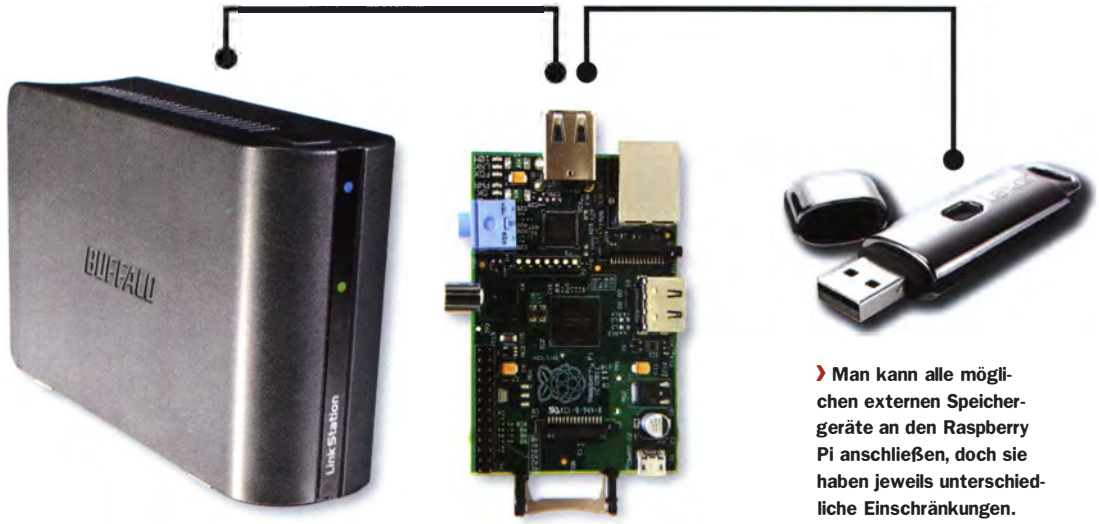
Da wir die SD-Karte belassen, wie sie von Raspbian eingestellt wurde, schauen wir uns einmal nur die mit **/dev/sd** Beginnenden an (Achtung, Sie finden nur Einträge, wenn Sie zusätzliche Speicher angeschlossen haben). Jedem Laufwerk wird ein Buchstabe, bei A beginnend, zugeordnet. **/dev/sda** ist demnach das erste Laufwerk, **/dev/sdb** das zweite usw. Zusätzlich wird jede Partition eines Laufwerks durchnummeriert. **/dev/sda1** ist somit die erste Partition auf dem ersten Laufwerk und **/dev/sdb3** ist die dritte Partition auf dem zweiten Laufwerk usw.

› Mit dem nützlichen Befehl df -h kann man sich anzeigen lassen, welche Laufwerke gerade an den Raspberry Pi angeschlossen sind.

Speicher hinzufügen

Man kann seinen Raspberry Pi um zusätzlichen Speicher in so ziemlich jeder Form ergänzen, sofern der neue Speicher über einen USB-Anschluss verfügt. USB-Memory-Sticks und externe

Speichergeräte



› Man kann alle möglichen externen Speichergeräte an den Raspberry Pi anschließen, doch sie haben jeweils unterschiedliche Einschränkungen.

Festplatten sind am gängigsten. Externe Festplatten sollten allerdings über eine eigene Stromversorgung verfügen, da der Raspberry Pi nicht dafür konzipiert ist, so viel Strom wie ein normaler Computer über USB zu liefern. Wenn die externe Festplatte also nur über einen USB-Anschluss und keinen externen Stromanschluss verfügt, wird sie wahrscheinlich nicht funktionieren. Mit einem USB-Hub mit externer Stromzufuhr kann man sich hier wiederum behelfen. Ein Speicherstick benötigt nicht so viel Strom und sollte über den USB-Anschluss des Pi funktionieren. Wenn Sie jedoch auch noch eine Maus und eine Tastatur verwenden möchten, benötigen Sie dennoch einen Hub, da es nur zwei USB-Anschlüsse gibt.

Hier müssen wir jedoch darauf hinweisen, dass die USB-Anschlüsse des RPi langsamer sind als bei den meisten Computern und man nicht die gleiche Übertragungsgeschwindigkeit erwarten darf. Besonders auffallen wird das bei USB 3.0-Geräten, die bei einem RPi nur das Leistungsniveau von USB 2.0 erreichen und darum weit langsamer funktionieren. Das ist der Nachteil, der in Kauf genommen werden musste, um das Board so klein und preiswert gestalten zu können.

Daten-Layout

Bisher haben wir das Wort Dateisystem verwendet, um damit die Verzeichnisstruktur auf dem Computer zu benennen. Es gibt jedoch noch eine weitere Bedeutung: die Art und Weise, wie Daten physisch auf einem Speichermedium gespeichert werden. Das ist wichtig, da jedes Laufwerk mit einem bestimmten Dateisystem formatiert werden muss, bevor es funktioniert. Es gibt eine Reihe verschiedener Dateisysteme: die einen werden von Windows verwendet, die anderen von Mac OS X, und die nächsten von anderen Systemen. Die gute Nachricht ist, dass Linux mit fast allen kompatibel ist. Wenn man also ein Speichermedium sowohl mit dem Pi als auch mit einem anderen Computer, auf dem kein Linux läuft, verwenden möchte, so formatiert man den Speicher am besten auf dem anderen Computer. Wenn man es andersherum macht, muss man darauf achten, den richtigen Dateisystemtyp für den Computer zu wählen. Die meisten Speichergeräte sind mit FAT32 vorformatiert, was von fast allen Geräten gelesen werden kann. Hierbei besteht allerdings das Problem, dass sehr große Dateien nicht verwaltet werden können (die Grenze liegt bei 4 GB). Wenn Sie jedoch das Speichermedium nur mit Ihrem Raspberry Pi und anderen Linux-Geräten verwenden wollen, dann erhalten Sie die beste Leistung, wenn Sie es mit einem Linux-Dateisystem formatieren. Das bekannteste Format ist ext4.

Wenn Sie das Speichergerät formatiert und angeschlossen haben, öffnen Sie ein Terminal und geben Sie folgenden Befehl ein:

```
df -h
```

Wurde eine Zeile ausgegeben? Wahrscheinlich nicht. Das liegt daran, dass das Gerät zuerst eingebunden werden muss.

Dies können wir im Terminal erledigen. Als Erstes benötigen wir einen „Mount-Point“. Das ist einfach ein Verzeichnis, das wir als „Eingangstür“ zu unserem Speichergerät benutzen. Es kann überall im Dateisystem liegen. Um die erste Partition des ersten Geräts in dem Verzeichnis **data** im Home-Bereich des Benutzers einzubinden, geben Sie folgende Befehle ein:

```
cd ~
mkdir data
sudo mount /dev/sda1 data
```

Natürlich wollen wir das nicht bei jedem Neustart unseres Pi wiederholen, besonders dann nicht, wenn wir ihn ohne Tastatur und Monitor verwenden möchten. Daher automatisieren wir die Durchführung. Es gibt eine Datei, die Linux vorgibt, was es wo einbinden soll. Diese heißt **fstab** und befindet sich in **/etc**. Um sie in einem einfachen Befehlszeilen-Texteditor zu öffnen, geben Sie im Terminal Folgendes ein:

```
sudo nano /etc/fstab
```

Die Datei beinhaltet eine Zeile für jede Partition auf jedem eingebundenen Gerät. Jede Zeile besteht aus einer Serie von durch einen Tabulator getrennten Werten, die dem System vorgeben, was es ausführen soll. Damit unser Speichergerät beim Neustart automatisch eingebunden wird, geben Sie in einer neuen Zeile am Ende von **fstab** Folgendes ein (Achtung, das Verzeichnis **data** muss wie im Beispiel zuvor bereits angelegt sein):

```
/dev/sda1       /home/pi/data
```

■

LXDE: Der Desktop

Linux hat nicht nur die Kommandozeile zu bieten. Sehen wir uns den grafischen Desktop, den Sie auf Ihrem Pi verwenden werden, einmal näher an.

Nun da Sie das Betriebssystem Raspbian installiert und zum Laufen gebracht haben, wird es Zeit, einen Blick auf Ihren Desktop zu werfen. Falls Sie bereits mit Windows (vor Version 8) oder Apples OS X gearbeitet haben, wird Ihnen als Erstes auffallen, wie vertraut alles aussieht. Es gibt eine Taskleiste, die am unteren Bildrand entlangläuft und Schnellstart-Icons für einige beliebte Anwendungen sowie ein Benachrichtigungsfeld mit einer Anzeige der momentan geöffneten Fenster enthält. Ganz links befindet sich das Haupt-Anwendungsmenü, ähnlich dem Startmenü von Windows, und rechts ist ein Applet-Bereich, wo unter anderem Uhr und Kalender angezeigt werden. Also alles in allem nichts Spektakuläres.

Was Sie ebenfalls feststellen werden, ist, wie schlicht alles gestaltet ist. Es gibt keine Schattenwürfe, Transparenzen oder 3D-Effekte, die auch eigentlich nicht notwendig sind und nur das Auge ablenken. Um solche Effekte darstellen zu können, müsste der Raspberry Pi mit einer leistungsstärkeren Hardware ausgestattet sein, was jedoch seine Herstellungskosten in die Höhe treiben würde.

Der Standard-Desktop von Raspbian heißt LXDE, was für Lightweight X11 Desktop Environment steht. Wie der Name suggeriert, ist LXDE tatsächlich ein Leichtgewicht in puncto benötigte Systemressourcen, und diese Eigenschaft macht es natürlich perfekt für den Raspberry Pi. In dem Namen steckt überdies, dass LXDE einer von vielen Desktops ist, die unter Linux zur Verfügung stehen. In der Tat gibt es etliche Alternativen, beispielsweise Xfce (dieser Desktop ist ein bisschen schicker gestaltet, und Sie könnten Ihn einmal ausprobieren, wenn Sie das mit mehr Speicher ausgestattete Modell B des Raspberry Pi ihr Eigen nennen) oder Mate (eine Mac-artige Benutzeroberfläche, die sich wachsender Beliebtheit erfreut), doch wir empfehlen, erst einmal bei LXDE zu bleiben.

> „Sie werden feststellen, wie schlicht alles gestaltet ist."

Doch sogar im Rahmen des LXDE-Desktops können Sie die Anzeigeoptionen in vielfältiger Weise anpassen – dies ist eine der Stärken der Linux-Philosophie: Wenn man irgendetwas verändern möchte, ist das in aller Regel möglich. LXDE können Sie mithilfe des Hauptmenüs anpassen, gehen Sie dort zu den Einstellungen. Sie können mithilfe einer Reihe von Optionen das optische Erscheinungsbild Ihres Desktops individuell anpassen. Am nützlichsten sind wahrscheinlich der „Openbox Configuration Manager" und das Menü „Customise Look and Feel".

Bei „Customise Look and Feel" können Sie Farben, Icons, Schriftarten und das Cursor-Verhalten verändern, während der „Openbox Configuration Manager" einem die Möglichkeit gibt, so ziemlich jeden Aspekt des Verhaltens von Programmfenstern bei LXDE zu steuern. Der Nutzer hat diesbezüglich weit mehr Freiheiten als bei den Desktops von Windows und OS X, wo sehr vieles vorgegeben ist.

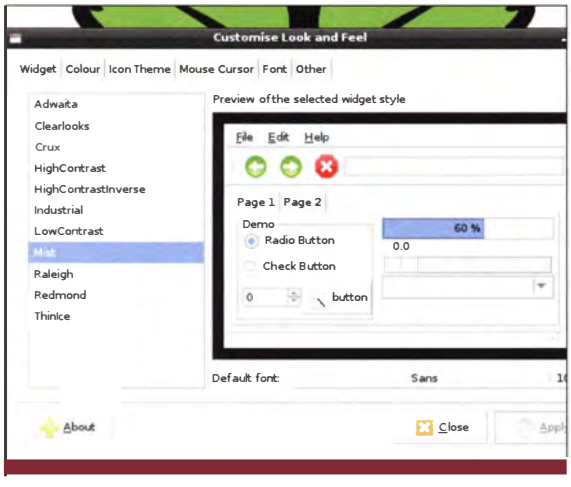
Dateimanager

Neben der grafischen Benutzeroberfläche beinhaltet LXDE noch ein weiteres sehr wichtiges Feature, nämlich den Dateimanager. Ähnlich dem Explorer bei Windows oder dem Finder bei OS X bietet er dem Anwender ein Werkzeug, um den Inhalt des Dateiensystems zu durchforsten, Verzeichnisse per Mausklick zu öffnen und Anwendungen zu starten.

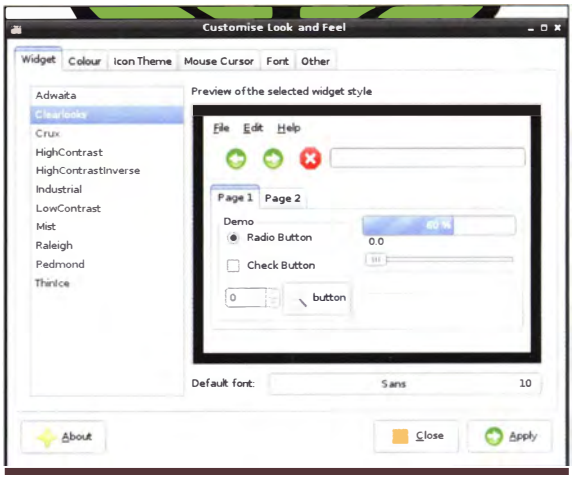
Im Hauptfenster des Dateimanagers werden die Dateien und Ordner angezeigt, die im aktuellen Verzeichnis liegen – der Pfad des aktuellen Verzeichnisses wird jeweils oben in der Adresszeile angezeigt. Mithilfe von Buttons lässt es sich bequem nach oben und unten sowie vor und zurück navigieren.

Im Bereich auf der linken Seite zeigt der Dateimanager auch Symbole an, welche die Geräte repräsentieren, die Sie an den Raspberry Pi angeschlossen haben.

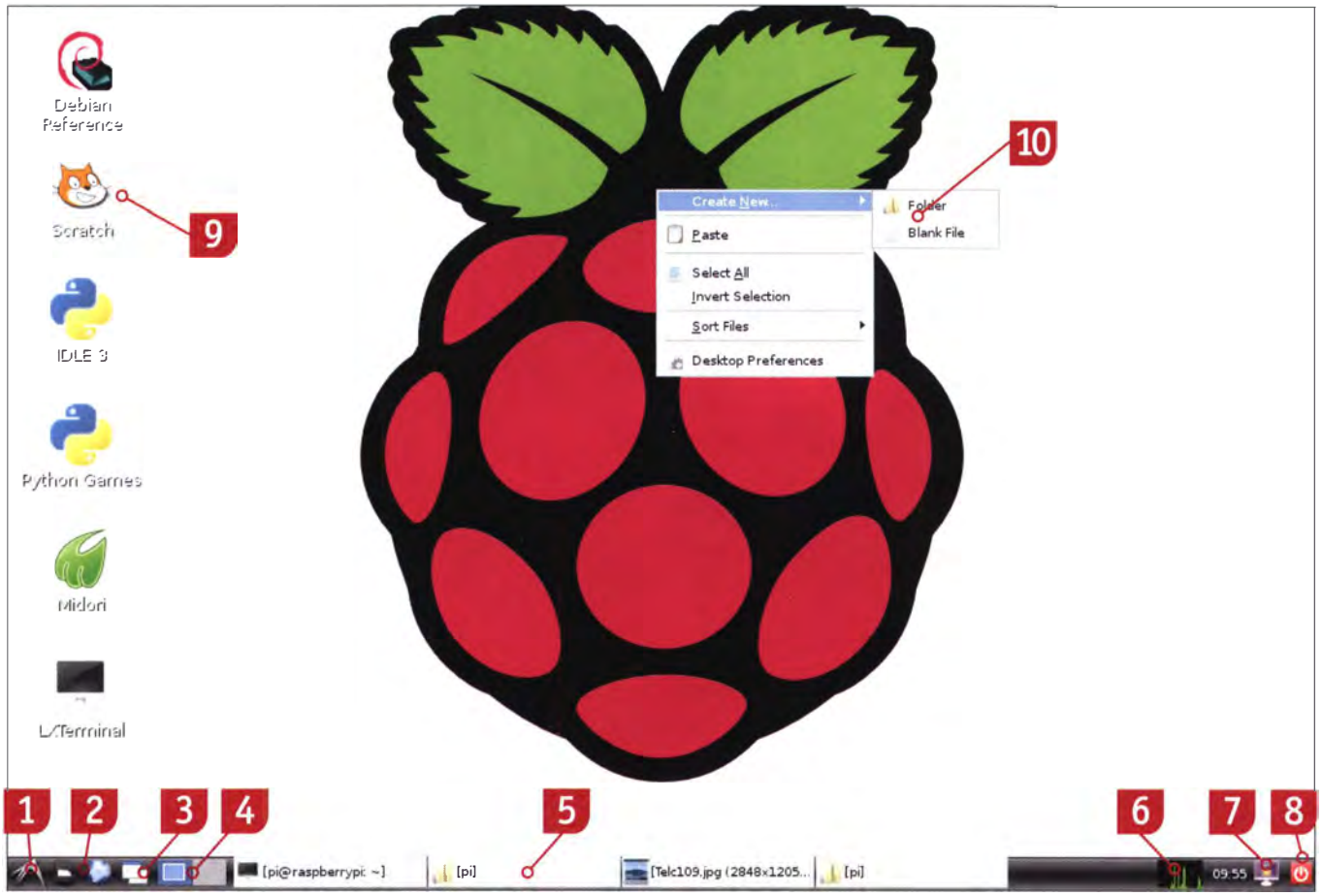
Auf der folgenden Seite finden Sie eine detaillierte Übersicht über die einzelnen Elemente der grafischen Benutzeroberfläche LXDE.



Falls Ihnen das Standarddesign von LXDE nicht zusagt, ändern Sie die Anzeigeoptionen im Einstellungsbereich des Hauptmenüs.



Dort werden mehrere Designs angeboten, unter denen Sie wählen können. Uns gefiel das moderner gestaltete „Clearlooks"-Thema recht gut.



1 LXDE-Hauptmenü

Das Hauptmenü entspricht dem Startmenü bei Windows und dem Hauptmenü bei OS X. Hier finden Sie Menüeinträge für alle Anwendungen, die standardmäßig in Raspbian enthalten sind, ebenso wie für die Programme, die Sie selber installiert haben (siehe Seite 36). Aus dem Haupmenü heraus können Sie außerdem den Raspberry Pi herunterfahren, Kommandos eingeben und das Raspbian-Einstellungsmenü aufrufen.

2 Schnellstartbereich

Hier können sich mehrere Icons befinden, in unserem Fall sind es zwei: Beim Klick auf das linke Icon öffnet sich ein Fenster des Dateimanagers, mit dem Sie durch Ihre Verzeichnisse und Dateien navigieren können. Das rechte Icon gehört zum Webbrowser *Dillo* (siehe auch Seite 38). Falls Sie stattdessen lieber den Browser *Midori* hier angezeigt bekommen möchten oder noch irgendwelche anderen Programme, dann machen Sie einen Rechtsklick auf den Schnellstartbereich und wählen Sie im daraufhin erscheinenden Menü den obersten Punkt – Settings – aus. Dies öffnet ein zweigeteiltes Fenster: rechts werden alle Anwendungen angezeigt, die derzeit auf Ihrem Raspberry Pi installiert sind, und links diejenigen, die im Schnellstartbereich verlinkt sind. Zwischen den beiden Hälften befinden sich Buttons, mit denen Sie Programme dem Schnellstartbereich hinzufügen oder aus diesem entfernen können.

3 Fenster verbergen

Mit einem Klick auf dieses Icon können Sie alle aktuell geöffneten Fenster auf einmal verbergen und so den Desktop freilegen.

4 Virtuelle Desktops

Wenn Sie gleichzeitig viele Fenster von Anwendungsprogrammen geöffnet haben, kann es schon mal recht voll werden auf Ihrem Desktop. Um dennoch die Übersicht zu behalten, können Sie sogenannte virtuelle Desktops einrichten. Diese rufen Sie mit einem Klick auf das entsprechende Icon auf. Auf einem virtuellen Desktop können Sie weitere Anwendungsfenster öffnen, während die Fenster des ursprünglichen Desktops ebenfalls geöffnet bleiben. Sie können mehrere virtuelle Desktops einrichten und zwischen allen Desktops ganz einfach per Mausklick hin und her springen.

5 Benachrichtigungsfeld

Alle momentan geöffneten Fenster samt den minimierten werden in diesem mittleren Bereich angezeigt. Sie können mit einem Mausklick auf eine der Flächen direkt zum entsprechenden Fenster springen. Die Bezeichnungen minimierter Fenster werden in eckigen Klammern angezeigt: [so].

6 CPU-Monitor

Dieses kleine Diagramm im Grün-auf-Schwarz-Retrolook zeigt die Aktivität des Prozessors Ihres Raspberry Pi an.

7 Bildschirmsperre

Sie können Ihren Desktop gegen unbefugte Zugriffe sperren, indem Sie es mit einer Passwortabfrage belegen. Denken Sie aber daran, dass Sie für derartige Zwecke das Standardpasswort des Geräts durch ein eigenes ersetzen sollten, denn das Passwort „raspberry“ ist natürlich jedem Pi-Anwender bekannt.

8 Ausschalten

Dieses Symbol suggeriert, Sie könnten Ihren Raspberry Pi damit herunterfahren. Stimmt aber nicht! Wenn Sie den Pi ordentlich ausschalten möchten und nicht bloß durch das Ziehen des Netzsteckers, dann öffnen Sie ein Terminal und tippen Sie das Kommando **sudo shutdown -h now** ein.

9 Icons der Anwendungen

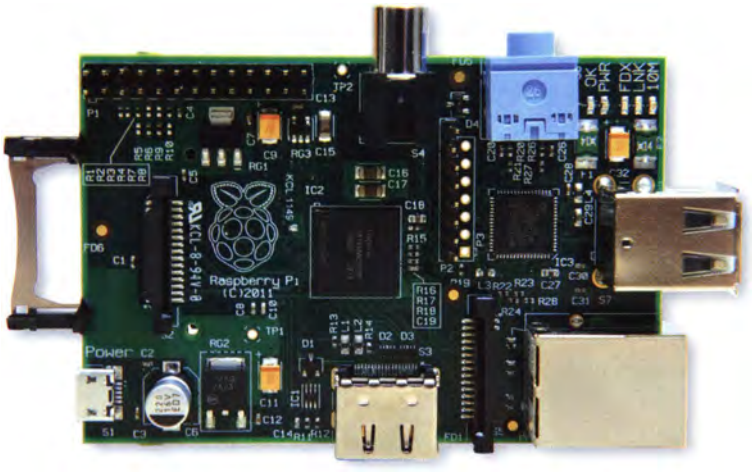
Wie bei den grafischen Benutzeroberflächen von Windows und OS X können Sie auf dem Desktop Ihres Raspberry Pi ganz nach Wunsch Icons mit Verknüpfungen zu bestimmten Programmen anordnen.

10 Kontextmenü

Dieses rufen Sie auf, indem Sie an einer beliebigen Stelle des Desktops einen Rechtsklick mit der Maus ausführen. ■

Pakete: Software

Erfahren Sie hier, wie der Paketmanager von Raspbian, apt-get, von Online-Repositorys Software herunterlädt und diese auf Ihrem System verwaltet.



Falls Sie an das Windows-Betriebssystem gewöhnt sind, ist es für Sie vermutlich selbstverständlich, dass jede Software ihren eigenen Installer mitbringt, der die richtigen Dateien an die richtigen Orte kopiert. Linux arbeitet jedoch normalerweise nach einem anderen Prinzip. Unter Linux gibt es eine Systemkomponente, die man gemeinhin als Paketmanager bezeichnet. Der Paketmanager holt und verwaltet die Software, die Sie brauchen. Er verlinkt auf ein Repository („Lagerstätte"), von wo er alle Programme herunterlädt. Da Linux auf Open-Source-Software baut, werden Sie fast alles, was sie benötigen, kostenlos in diesen Repositorys finden. Sie müssen sich auch nicht darum kümmern, wo genau Sie die Installationsdateien finden: Das alles erledigt der Paketmanager für Sie.

Es gibt mehrere Paketmanager für Linux. Raspbian benutzt **apt-get,** Arch hingegen einen anderen. Falls Sie also letztere Distribution auf Ihrem Raspberry Pi ausprobieren möchten, müssen Sie sich mit der Software **pacman** auseinandersetzen, die wir hier aber nicht behandeln.

Bevor wir anfangen, weisen wir darauf hin, dass Sie Ihren Raspberry Pi mit dem Internet verbinden müssen. Da wir Software aus dem Internet herunterladen werden, ist eine Verbindung zum Internet für das Befolgen dieser Anleitung zwingend notwendig.

apt-get ist ein Kommandozeilenprogramm, für das Sie ein Terminal verwenden müssen (mehr zur Kommandozeile auf Seite 24). Da die Verwaltung von Softwarepaketen das gesamte System betrifft, müssen alle Befehle mit einem vorangestellten **sudo** ausgeführt werden. Stellen Sie zunächst sicher, dass Sie die aktuelle Software in Ihrer Paketliste zur Verfügung haben. Führen Sie dazu folgenden Befehl aus:

```
sudo apt-get update
```

Da die gesamte Software über den Paketmanager verwaltet wird, kann er alle Programme auf dem Rechner aktualisieren. Sie müssen sich also nicht für jede Anwendung einzeln damit befassen. Um alle Updates für die gesamte installierte Software herunterzuladen und zu installieren, verwenden Sie den Befehl:

```
sudo apt-get upgrade
```

Das könnte ein Weilchen dauern, da Open-Source-Software dazu neigt, recht oft aktualisiert zu werden.

Bei Linux installiert man – um in der Terminologie zu bleiben – keine Anwendungen, sondern sogenannte Pakete. Diese bestehen jeweils aus einem Satz Dateien, der zwar meistens einem Anwendungsprogramm entspricht, jedoch nicht immer. Ebenso gut kann es sich bei einem solchen Paket um Daten für ein Spiel, eine Dokumentation oder ein Plug-in handeln.

Um mit **apt-get** Software zu installieren, müssen Sie den Namen des Pakets kennen, das Sie installieren möchten. Meist ist dieser offensichtlich, er muss aber vollständig richtig sein, damit das Paket heruntergeladen werden kann. Falls Sie sich nicht sicher sind, können Sie alle zur Verfügung stehenden Pakete mit **apt-cache** durchsuchen. Probieren Sie beispielsweise folgenden Befehl aus:

```
apt-cache search iceweasel
```

Diese Eingabe wird eine Liste von Paketen auswerfen, die mit dem gesuchten Webbrowser (*Iceweasel* ist eine spezielle Version von *Firefox*) zu tun haben. Um *Iceweasel* zu installieren, verwenden Sie folgendes Kommando:

```
sudo apt-get install iceweasel
```

Sie werden feststellen, dass **apt-get** Sie um die Zustimmung zur Installation einer Anzahl von Paketen fragt. Dabei handelt es sich um die sogenannten Abhängigkeiten. *Iceweasel* benötigt den Inhalt dieser Pakete, um zu funktionieren. Normalerweise müssen Sie sich über die angezeigten Abhängigkeiten nicht den Kopf zerbrechen – bestätigen Sie deren Installation einfach nur mit „Y", und der Paketmanager wird alles andere für Sie erledigen. Allerdings gibt es Pakete, die eine große Anzahl von Abhängigkeiten aufweisen und somit viel Speicherplatz benötigen. Falls Ihre SD-Karte bereits relativ voll ist, sollten Sie gegebenenfalls vor der Installation Platz schaffen.

Falls Sie *Iceweasel* und alle zugehörigen Pakete (sofern diese nicht von anderen Paketen benötigt werden) wieder deinstallieren möchten, erledigen Sie dies mit dem Befehl:

```
sudo apt-get purge iceweasel
```

Oft begegnen Sie Paketen, die ein **-dev** am Ende ihres Namens aufweisen. Diese brauchen Sie nur, wenn Sie planen, die Software selbst zu kompilieren. Normalerweise können Sie solche Pakete ignorieren. **apt-get** ist ein großartiges Werkzeug, lässt jedoch etwas an Benutzerfreundlichkeit zu wünschen übrig. Es gibt aber auch grafische Tools zur Paketverwaltung. Zwei davon sind *Synaptic* sowie der Raspberry Pi App Store (siehe auch Seite 37). *Synaptic* funktioniert wie **apt-get**, hat aber eine grafische Benutzeroberfläche. Sie können das Programm mit **sudo apt-get install synaptic** installieren und mit **--where is synaptic** starten. Im Raspberry Pi App Store finden Sie nicht nur freie Software wie mit **apt-get** und *Synaptic*, sondern auch kommerzielle Pakete, die Sie kaufen müssen. Der App Store ist auf Raspbian vorinstalliert und kann über das entsprechende Icon auf dem Desktop aufgerufen werden.

sicher im Griff

Weitere Informationen

Synaptic

Mit *Synaptic* können Sie alles erledigen, was Sie auch mit dem Kommando **apt-get** tun können, allerdings ist das grafische Interface etwas einfacher zu verwenden. Das zeigt sich vor allem beim Suchen von Paketen, da das Fenster die Ergebnisse übersichtlicher als das Terminal aufbereitet.

Raspberry Pi App Store

Der Raspberry Pi App Store erlaubt es den Anwendern, die Software zu bewerten. So sehen Sie sofort, wie gut sie von anderen Nutzern eingeschätzt wird. Allerdings hat der App Store bei weitem nicht den Umfang von **apt-get** oder *Synaptic*.

Software kompilieren

Manchmal benötigen Sie Pakete, die Sie nicht in einem Repository finden und damit auch nicht mit **apt-get** installieren können. In diesem Fall müssen Sie die Software selbst kompilieren. Verschiedene Projekte verpacken ihre Software auf verschiedenen Arten, aber normalerweise funktioniert folgende Anleitung: Laden Sie sich den Quellcode von der Website des Projekts herunter und entpacken Sie ihn. Meist endet der Dateiname mit .tar.gz oder .tgz. Falls das zutrifft, entpacken Sie die Datei mit:

```
tar zxvf <dateiname>
```

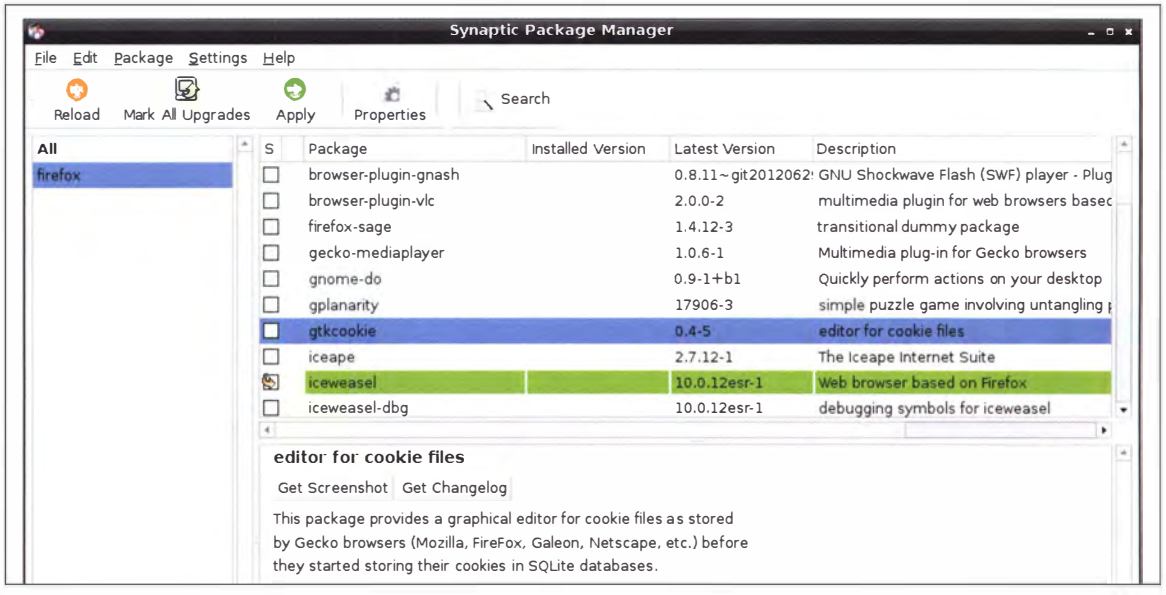
Falls der Name der Datei mit .tar.bzip2 endet, ersetzen Sie zxvf durch xjf. Das sollte ein neues Verzeichnis anlegen, das Sie mit **cd** aufrufen. Hier hoffen wir darauf, dass eine Datei mit dem Namen **INSTALL** zu finden ist, die Sie mit **less INSTALL** lesen können. Das sollte Ihnen alles Nötige verraten, um mit der Installation fortfahren zu können. Oft (aber nicht immer) ist der Weg der folgende:

```
./configure
make
sudo make install
```

Die erste Zeile überprüft, ob auf Ihrem System alle notwendigen Abhängigkeiten vorhanden sind. Falls die Überprüfung fehlschlägt, müssen Sie sicherstellen, dass alle nötigen **-dev**-Pakete vorhanden sind.

Falls sich alle Befehle ohne Fehler ausführen lassen, ist die Software installiert und bereit zur Verwendung. Dieser Weg ist aber wesentlich fehleranfälliger als die Benutzung eines Paketmanagers. ■

› **apt-cache in einer Kommandozeile zeigt Ihnen eine Liste aller erhältlichen Pakete an.**



› ***Synaptic* stellt Ihnen ein einfach zu verwendendes Front-End für den apt-Paketmanager zur Verfügung.**

Apps: Standard-

Werfen wir einen Blick ins Innere des Raspberry Pi – und auf die Software, die aus diesem kleinen Gerät eine Sahneschnitte macht.

Webbrowser

Midori ist ein genügsamer Webbrowser (klicken Sie auf das grüne Icon auf dem Desktop, um ihn zu starten). Mit „genügsam“ ist hier gemeint, dass der Browser im Vergleich zu *Firefox, Internet Explorer* oder *Apples Safari* sehr wenig Systemressourcen benötigt, weswegen er optimal für den Raspberry Pi geeignet ist. Ungeachtet seiner geringen Größe bringt das Programm viele Features mit, die auch seine größeren Konkurrenten haben.

Midori basiert auf der WebKit-Engine und liefert somit die gleichen HTML-Rendering-Ergebnisse wie *Safari* oder Googles Browser *Chrome*. Außerdem ist *Midori* klug genug zu erkennen, wann Sie einen Suchbegriff in die URL-Leiste eingeben. Anstatt Ihnen einfach zu sagen, Sie hätten einen Fehler gemacht (wie bei Safari der Fall), erkennt Midori, dass Sie nach etwas suchen, und leitet den Begriff an seine Standard-Suchmaschine weiter. Das ist ausnahmsweise nicht Google, sondern Duck Duck Go. Diese Suchmaschine durchforstet vor allem Crowdsourcing-Websites à la Wikipedia und liefert daher etwas andere Treffer als Google – was im Vergleich mal bessere, mal schlechtere Ergebnisse zeitigt. Was uns an Duck Duck Go am besten gefällt, ist die Tatsache, dass es Ihre Suchanfragen nicht speichert. Es respektiert Ihre Privatsphäre, was im Web heutzutage selten ist. Geben Sie Duck Duck Go eine Chance, und wenn Sie nicht zufrieden sind, können Sie jederzeit zu Google oder Yahoo wechseln, indem Sie das Enten-Icon links von der Suchleiste anklicken.

> „Midori benötigt sehr wenig Systemressourcen, bringt jedoch viele Features mit.“

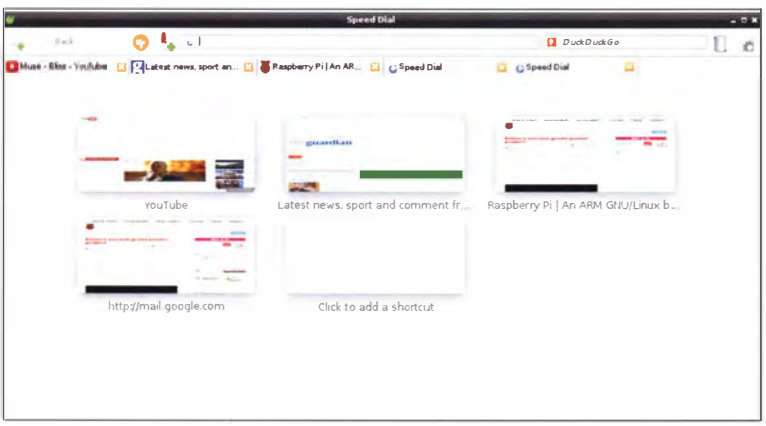
Ein weiteres tolles *Midori*-Feature ist seine Schnellwahl. Wenn Sie eine neue Registerkarte über das Tastaturkürzel STRG+T öffnen, sehen Sie ein kleines, leeres Fenster. Klicken Sie auf „Click to Add Shortcut“ („Klicken, um Shortcut hinzuzufügen“), tippen Sie eine oft besuchte URL in die Leiste ein (etwa **mail.google.com** oder **twitter.com**), und beim nächsten Öffnen einer Registerkarte sehen Sie ein Vorschaubild der entsprechenden Seite und können diese mit nur einem Klick laden. Das ist ein nettes Feature, das ursprünglich für einen anderen Linux-Browser, *Opera*, entwickelt wurde und danach beim *Firefox* und vielen anderen beliebten Browsern übernommen wurde.

Andere Browser

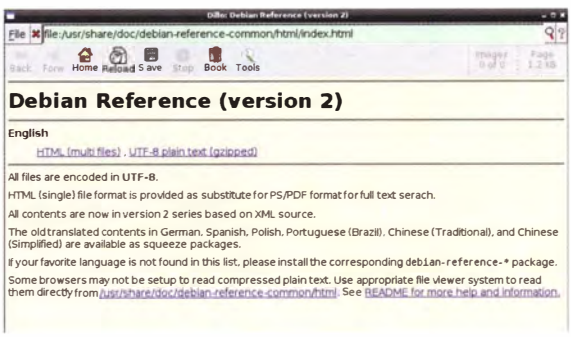
Midori ist nicht der einzige in Raspbian enthaltene Webbrowser (obwohl er unserer Meinung nach der beste ist), sondern Sie bekommen noch einen weiteren Browser namens *Dillo*. Wenn Sie diesen öffnen, wird Ihnen auffallen, dass er ziemlich altertümlich aussieht. Der Grund dafür ist, dass *Dillos* grafisches Interface mithilfe eines alten Toolkits von vor der Jahrtausendwende entwickelt wurde.

Solch eine alte Software zu benutzen, bringt allerdings auch einen Vorteil mit sich: Da die Rechner damals nicht besonders leistungsstark waren, lädt *Dillo* mit moderner Hardware extrem schnell – selbst auf einem Raspberry Pi fühlt sich der Browser ziemlich flink an. Die Kehrseite der Medaille ist allerdings, dass *Dillo* nicht besonders gut funktioniert. Als er entwickelt wurde, war JavaScript nicht einmal ansatzweise so weit verbreitet wie heute. Daher hat der Browser jedes Mal zu kämpfen, wenn er mit Webseiten konfrontiert wird, die JavaScript benutzen.

Unter dem Internet-Menü sehen Sie außerdem etwas, das sich *Midori Private Browsing* nennt. Das ist im Grunde die gleiche Anwendung wie *Midori*, allerdings öffnet sie ein Fenster im Inkognito-Modus, bei dem der Browserverlauf nicht gespeichert wird.



› **Die Schnellwahl-Option von *Midori* ermöglicht es Ihnen, Ihre Lieblingsseiten mit einem Klick aufzurufen.**



› **Mit einem Doppelklick auf das Debian Reference-Icon auf dem Desktop öffnen Sie die Datei in *Dillo*. Die Anzeige ist dabei ziemlich altertümlich.**

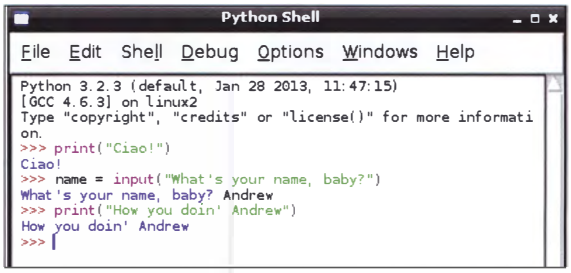
Software

Code schreiben

Da sich der Raspberry Pi an die kommende Generation von Programmierern richtet, ist es nicht weiter verwunderlich, dass die Programmieranwendungen ganz prominent in der Standard-Software-Auswahl von Raspbian platziert sind.

Als Erstes wäre hier IDLE zu nennen. IDLE ist eine Entwicklungsumgebung für die Programmiersprache Python. Natürlich ist es auch möglich, Code mithilfe der Kommandozeile zu schreiben, aber eine Entwicklungsumgebung kann den Prozess deutlich angenehmer gestalten. IDLE beinhaltet eine automatische Code-Vervollständigung, um Sie vor lästigen Vertippern zu bewahren, eine Syntaxhervorhebung, sodass verschiedene Elemente des Codes auf dem Display in verschiedenen Farben dargestellt werden, und einen Debugger, um Fehler leichter zu finden. Zwar kann IDLE Ihnen nicht das Denken abnehmen, aber falls Sie die schließende Klammer am Ende einer Anweisung vergessen haben sollten und Ihr Code darum nicht ordentlich funktioniert, wird IDLE den Fehler für Sie markieren und Ihnen so langes Suchen ersparen.

Zugegeben, IDLE und IDLE 3 könnten besser bezeichnet sein, denn es handelt sich dabei um das gleiche Programm. Der einzige Unterschied: IDLE 3 ist ein Teil von Python 3, der neuesten Version der Programmiersprache Python, während IDLE in Version 2 enthalten ist. Da Raspbian beide Python-Versionen beinhaltet, sind beide auf dem Desktop platziert. Wenn Sie nicht gerade eines der neuesten Features von Python ausprobieren wollen, empfehlen wir Ihnen IDLE und Python 2, denn das ist der Standard bei Raspbian.



› Farbige Hervorhebungen (wie hier in IDLE) erleichtern es Ihnen, Fehler in Ihrem Code zu entdecken.

Scratch

Auf Seite 46 in diesem Heft lesen Sie alles über Scratch, weswegen wir uns hier auf eine knappe Einführung beschränken. Es handelt sich um eine Entwicklungsumgebung ähnlich IDLE, aber anstatt den Code über die Tastatur einzutippen, ziehen Sie hier Codezeilen aus einer Liste auf der linken Seite zum Programmierbereich in der Mitte des Bildschirms. Das Ziel ist es, einer animierten Katze Anweisungen zu geben.

Das alles mag jetzt ein wenig albern klingen, aber wenn Sie dem Programm eine Chance geben, werden Sie feststellen, wie intuitiv es ist und wie gut es Kindern oder anderen Neulingen in dem Bereich die Programmierkonzepte näherbringt. Sie werden Scratch wohl nie im professionellen Bereich nutzen, aber Schleifen, Listen und bedingte Anweisungen sind das A und O beim Lösen von Programmierproblemen, unabhängig von der von Ihnen benutzten Sprache.

Auf dem Raspberry Pi ist eine weitere Programmiersprache namens Squeak installiert, aber wir empfehlen Ihnen, bei Scratch und Python zu bleiben.

› Scratch bietet eine grafische Darstellung der Schlüsselkonzepte des Programmierens.

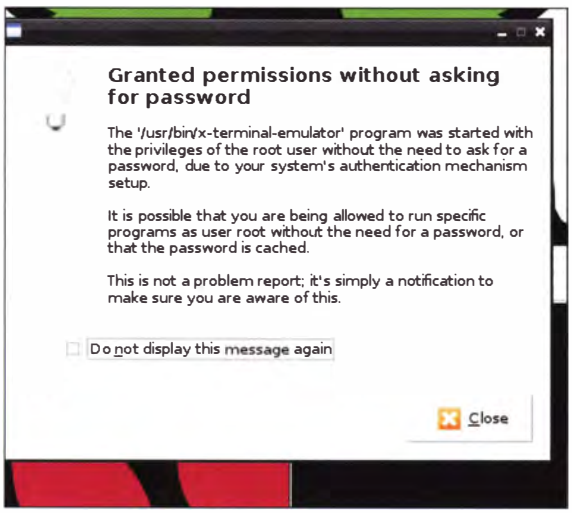
Warum Python?

Es gibt viele Programmiersprachen, doch die Schöpfer des Raspberry Pi und des Raspbian-Betriebssystems haben Python als Basis ihrer Tutorials benutzt, und das aus drei guten Gründen:
1: Python ist einfach zu lesen. Nachdem Ihnen eine Zeile des Python-Codes erklärt wurde, ergibt sie sofort einen Sinn. Der Befehl **print("Hello world!")** zum Beispiel lässt die Wörter „Hello world!" auf dem Bildschirm erscheinen. Das ist zwar ein sehr einfaches Beispiel, aber es zeigt, dass Python sehr gut lesbar sowie leichter zu verstehen ist als andere Sprachen.
2: Python ist eine interpretierte und keine kompilierte Sprache. Bei einer kompilierten Sprache wie C oder C++ muss der Code erst geschrieben und dann mithilfe eines separaten Programms namens Compiler in Nullen und Einsen gewandelt werden. Erst danach kann man den Code laufen lassen, sodass Sie auch erst dann herausfinden können, ob Ihnen ein Fehler unterlaufen ist. Python ist eine interpretierte Sprache, sodass sie direkt vom Interpreter läuft.
3: Python ist plattformübergreifend. Sowohl Python als auch der Interpreter sind für Linux, Mac OS X und Windows verfügbar, sodass Sie an jedem Rechner arbeiten können, wenn Sie die Sprache kennen.

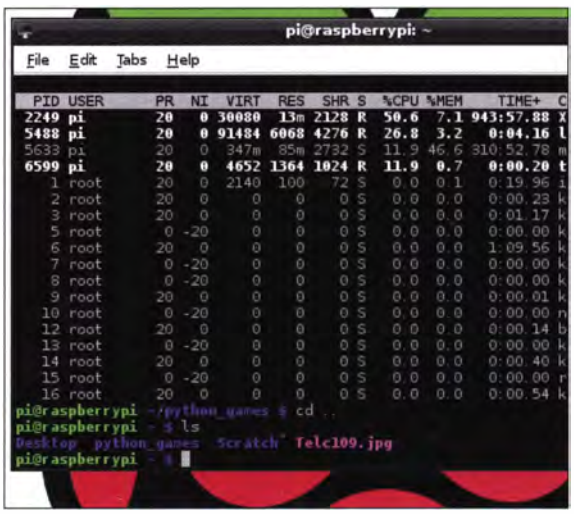
Kommandozeile

Wie wir auf Seite 24 festgestellt haben, gibt Ihnen die Kommandozeile eine Menge Kontrolle über die Funktionsweise Ihres Computers. Die Kommandozeile öffnen Sie mit den Befehlen STRG+ALT+F1 bis STRG+ALT+F6 (mit STRG+ALT+F7 kehren Sie zur grafischen Benutzeroberfläche zurück). Wenn Sie aber die grafische Umgebung beibehalten wollen – etwa, weil Sie Befehle von einer Webseite kopieren –, dann können Sie die Konsolenanwendung auf dem Desktop benutzen, um in die Kommandozeile zu gelangen.

Außerdem sehen Sie im Zubehörmenü etwas, das sich Root Terminal nennt. Damit öffnen Sie im Prinzip eine normale Konsole, sind allerdings als Root-User angemeldet. Als Root-User haben Sie Berechtigungen über das ganze System und nicht nur über die Benutzerdateien, sodass ein Ausrutscher auf der Tastatur für große Schwierigkeiten sorgen kann. Daher raten wir Ihnen davon ab, die Konsole in diesem Modus zu benutzen. Wenn Sie noch nicht ganz überzeugt sind, schauen Sie sich bitte den Kasten über Root-Berechtigungen auf Seite 26 an.



› Lassen Sie Vorsicht walten, wenn Sie sich als Root-User angemeldet haben. Noch besser: Tun Sie es gar nicht erst, falls es nicht unbedingt nötig ist.



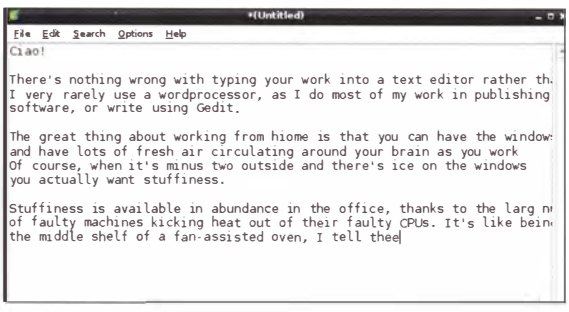
› Die Konsole bietet Ihnen eine bequeme Möglichkeit, unter die grafische Benutzeroberfläche zu schauen und zu sehen, was dort abläuft.

Zubehör

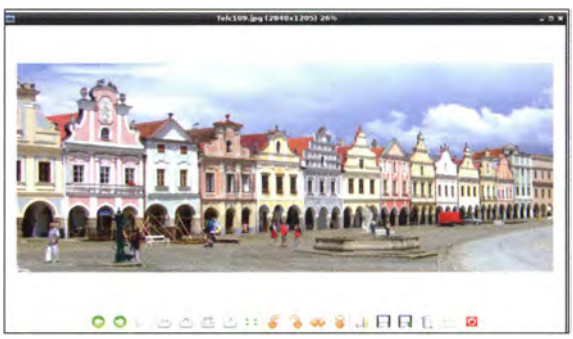
Große Textverarbeitungsprogramme wie *Microsoft Word* oder *LibreOffice* sind echte Software-Brocken. Sollten Sie versuchen, sie auf einem Raspberry Pi zum Laufen zu bringen, würde dieser mit seinem bescheidenen 700-MHz-Prozessor strampeln. Zum Vergleich: Der schwächste Apple iMac hat einen 2,7-GHz-Prozessor, der also immer noch fast viermal schneller ist. Anstelle eines Textverarbeitungsprogramms gibt Ihnen Raspbian *Leafpad*. Dabei handelt es sich um einen sehr einfachen Texteditor, der Formatierungen, Einbettungen von Bildern und das ganze andere überflüssige Zeug weglässt, an das wir uns über die Jahre so gewöhnt haben. Stattdessen konzentriert sich *Leafpad* darauf, Ihnen ein Fenster zu geben, in das Sie Wörter eintippen können. Das ist alles.

Mit einem Doppelklick auf eine Bilddatei öffnen Sie diese im Vollbildmodus im Dateimanager, aber wenn Sie sich die Zeit zur Eingewöhnung nehmen, werden Sie *Image Viewer* als viel praktischer empfinden. Das Programm kann Bilder rotieren und umdrehen sowie hinein- und herauszoomen, sodass es besser zur Ansicht einer Bildersammlung geeignet ist als der Standard-Dateimanager. Auch hier gilt: Das Programm hat nicht den Schnickschnack von *Adobe Photoshop* oder dessen Freeware-Alternativen, aber es erfüllt seinen Zweck adäquat und funktioniert gut mit der meisten Hardware.

Zu guter Letzt wäre noch *Xarchiver* zu nennen. Wenn Sie eine große Menge Dateien auf dem Raspberry Pi erstellen, wird Ihnen sehr bald der Speicher ausgehen. Die Lösung: Entweder Sie kaufen sich eine SD-Karte mit mehr Speicher, oder aber Sie verpacken Ihre Dateien mit *Xarchiver* und speichern Sie anderswo – auf einem anderen Rechner oder bei einem Cloud-Anbieter.



› Schreiben Sie etwas und erstellen Sie Dokumente mit *Leafpad* (auch wenn es sich nur um ganz schlichte Textdateien handelt).



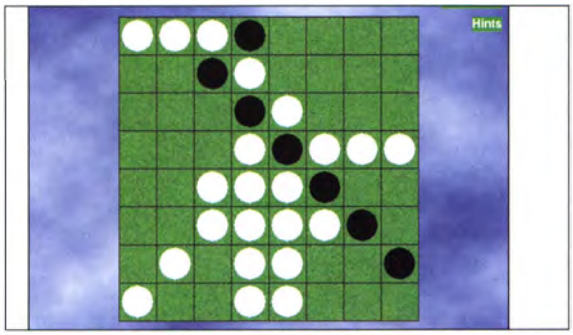
› Das eingebaute Programm zur Bildanzeige, *Image Viewer*.

Python-Spiele

Raspbian verfügt standardmäßig über eine kleine, aber feine Auswahl an Spielen. Wir haben zwar schon von Menschen gehört, die sogar *Quake III* auf dem Raspberry zum Laufen gebracht haben, aber die vorhandenen Spiele sind allesamt einfache Puzzle-Games für Kinder. Mit einem Doppelklick auf das Spiele-Icon auf dem Desktop öffnen Sie ein Menü mit zehn Games, allesamt Klone von berühmten Denkspielen, die es schon seit Jahren gibt, oder von klassischen Kinderspielen (etwa *Connect Four*, das hier als *Four in a Row* auftaucht).

Was noch besser ist: Alle Spiele sind quelloffen, sodass Sie freie Hand haben. So können Sie etwa mithilfe der Entwicklungsumgebung IDLE mit dem Quellcode herumexperimentieren und schauen, was passiert.

Auf dem Raspberry Pi ist auch *Starpusher* vorhanden – eine Kopie des klassischen japanischen Spiels *Sokoban*, eines Puzzlespiels mit Suchtfaktor. Es ist simpel, kann einen zur Verzweiflung treiben, aber auch sehr befriedigend sein. Dann ist da noch das rätselhafte *Squirrel Eat Squirrel* und – wie auf Tausenden von Mobiltelefonen – *Wormy*. Diese Nachahmung von *Snake* bietet immer noch eine der besten Möglichkeiten, eine Busfahrt zu verbringen.



Flippy

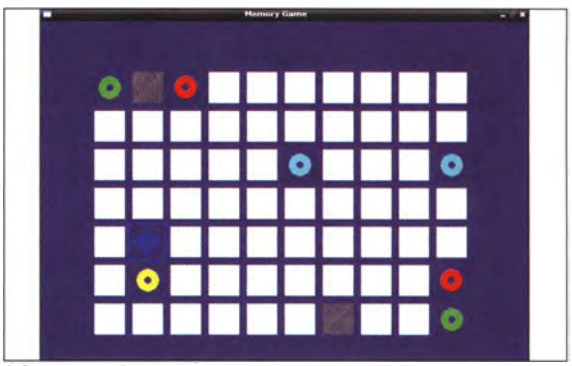
Othello (aus rechtlichen Gründen auch als *Reversi* bekannt) wird auf dem Raspberry Pi als *Flippy* wiedergegeben. Platzieren Sie Ihre Steine auf dem Brett so, dass diese die gegnerischen Steine umzingeln.

Four in a Row

Werfen Sie Steine in das Rastergitter, um eine Viererreihe zu bilden. Uns ist durchaus klar, dass dies ein Kinderspiel ist, aber es ist trotzdem äußerst befriedigend, den Computer darin zu besiegen.

gemgem

Eine Nachahmung des süchtig machenden *Bejeweled*, bei dem man benachbarte Steine vertauscht, um Reihen und Spalten aus zueinander passenden Steinen zu bilden. Je mehr Blöcke zusammenpassen, desto mehr Punkte bekommen Sie.

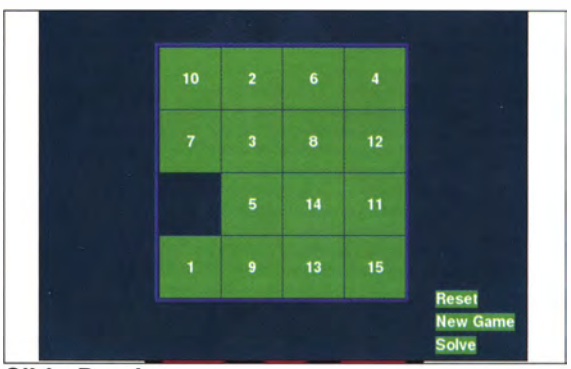


Memory Puzzle

Decken Sie die verschiedenen Icons auf und versuchen Sie, sich die zueinander passenden und deren Positionen zu merken. Extrem frustrierend für Farbenblinde und nur unwesentlich weniger frustrierend für alle anderen.

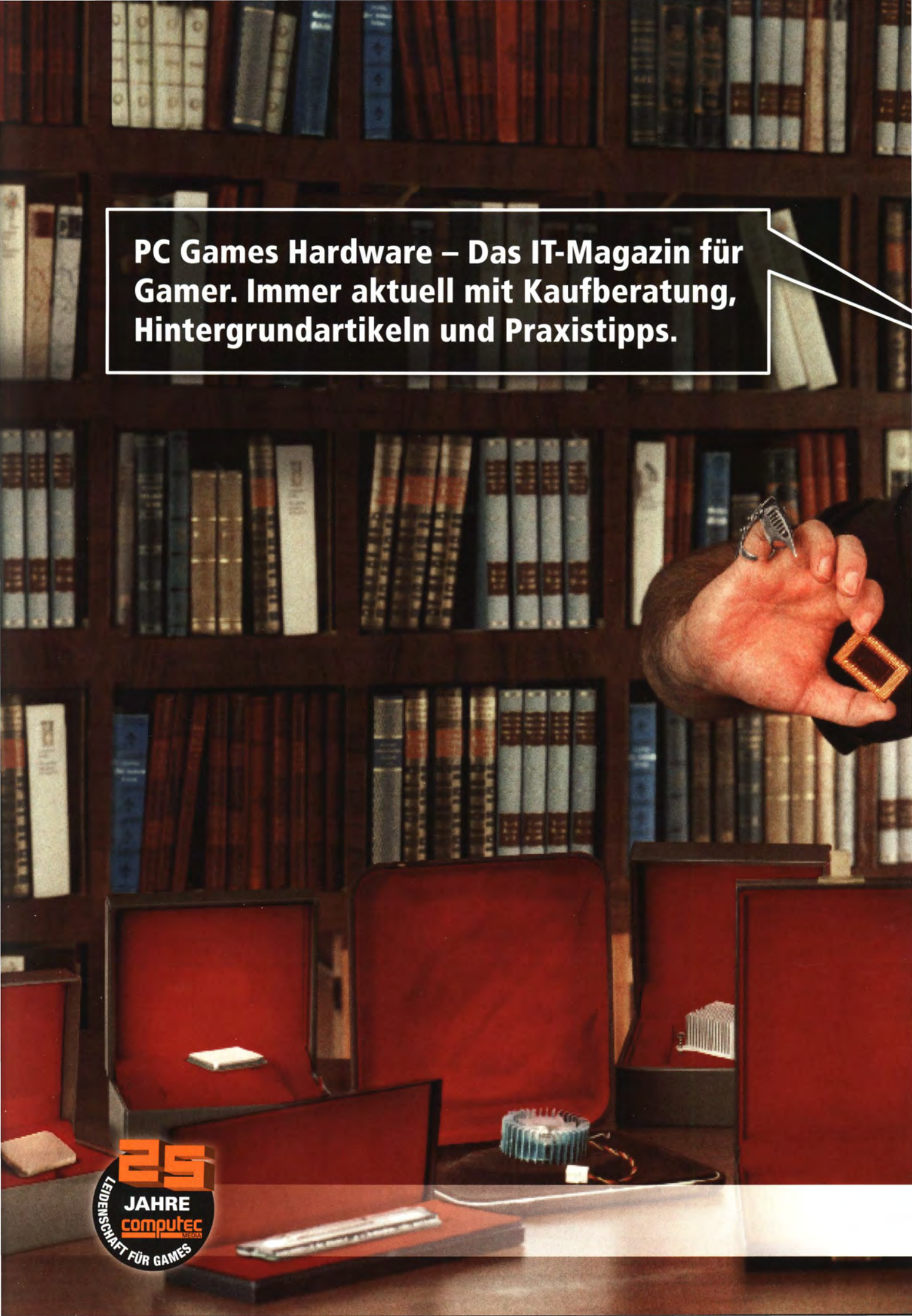
Simulate

Die vier Quadrate auf dem Bildschirm leuchten abwechselnd auf. Alles, was Sie tun müssen, ist es, sich die jeweilige Abfolge zu merken und danach durch Anklicken der Quadrate in der richtigen Reihenfolge zu wiederholen.

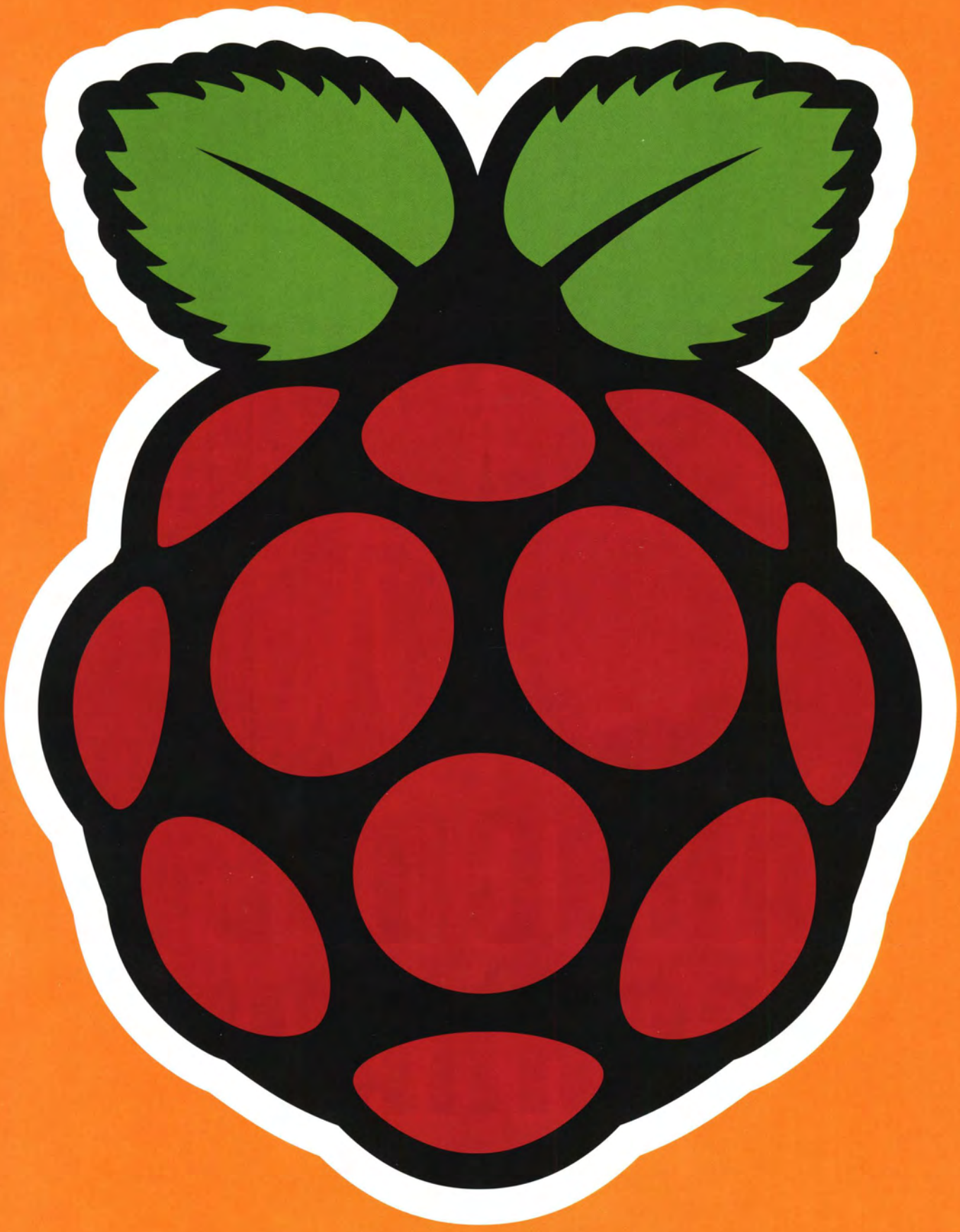


Slide Puzzle

Schieben Sie die Kacheln herum, um die richtige Nummernabfolge zu bilden. *Slide Puzzle* bietet wirklich fast gar nichts fürs Auge, eignet sich also perfekt dafür, von Ihnen modifiziert zu werden, wenn Sie sich erst einmal mit Python auskennen.

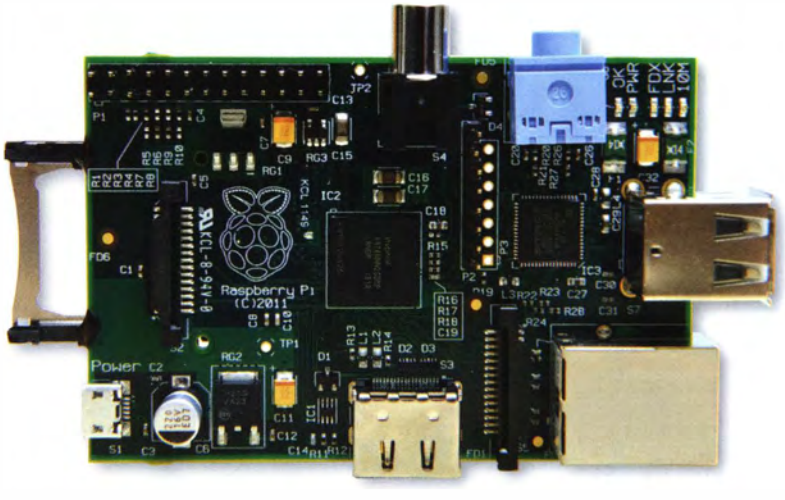

PC Games Hardware – Das IT-Magazin für Gamer. Immer aktuell mit Kaufberatung, Hintergrundartikeln und Praxistipps.
25
JAHRE
computec
MEDIA
LEIDENSCHAFT FÜR GAMES


PC Games Hardware
www.pcgameshardware.de • www.pcghx.de
DAS HARDWARE-MAGAZIN FÜR PC-SPIELER
HARDCORE FÜR SCHRAUBER
QR-Code scannen und hinsurfen!
Neue Ausgabe jetzt am Kiosk erhältlich oder einfach online bestellen unter: www.pcgh.de/shop
Auch erhältlich als ePaper bei:
computec MEDIA KIOSK
iKiosk
Online-Kiosk
PresseKatalog Online-Kiosk



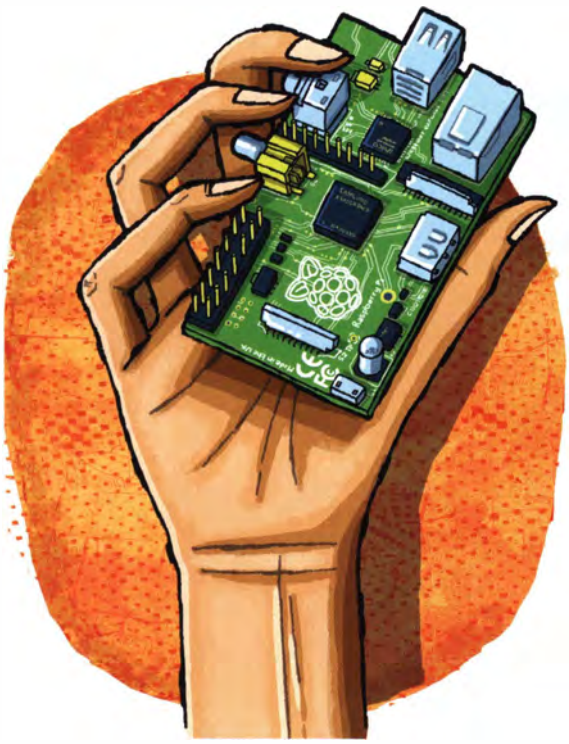
Tutorials

Der Raspberry Pi eignet sich für ein riesiges Spektrum von Anwendungen, der Phantasie sind dabei kaum Grenzen gesetzt. In diesem Teil des Handbuchs stellen wir einige davon vor, wobei wir uns sowohl den Software- als auch den Hardware-Aspekt näher anschauen.



Programmieren:

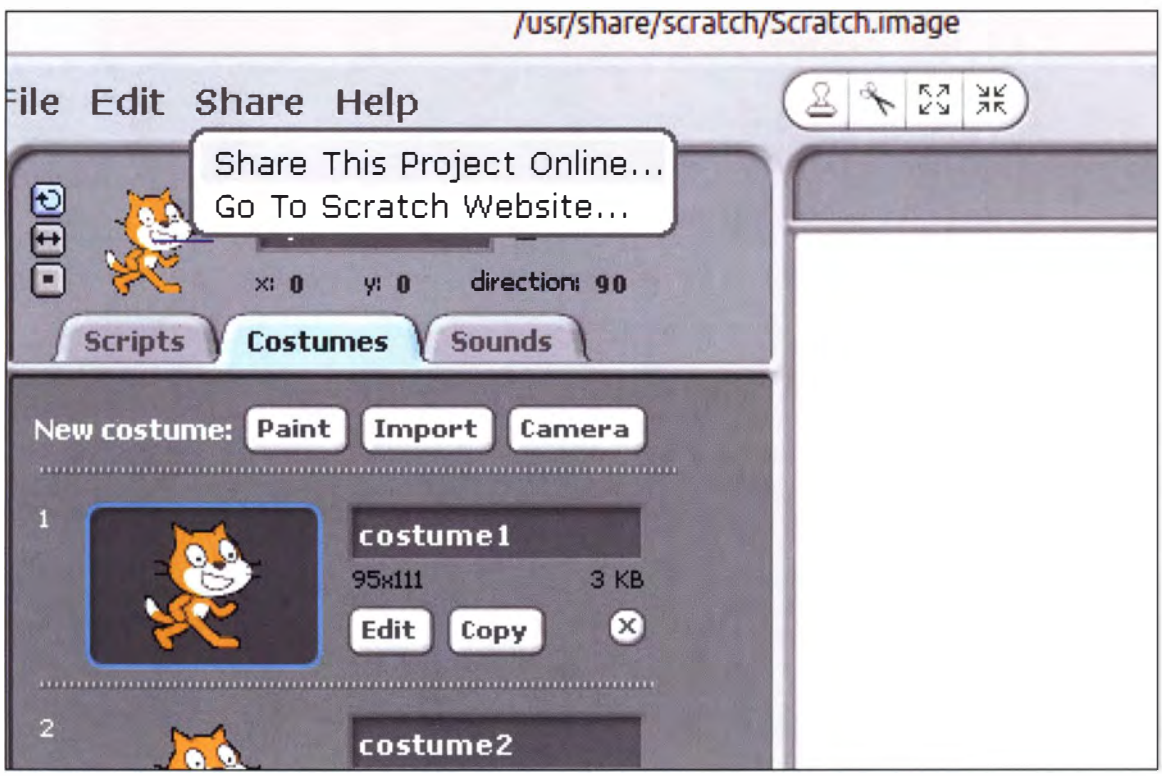
In diesem Beitrag zeigt Ihnen Ben Everard, wie Sie auf dem Raspberry Pi ein einfaches Katz-und-Maus-Spiel programmieren können.



Mit dem Raspberry Pi können Sie ein breites Spektrum von Programmiersprachen verwenden. Eine davon heißt Scratch. Für Anfänger ist Scratch eine tolle Sprache, denn sie führt viele grundlegende Mechanismen des Programmierens ein und ist dabei gleichzeitig sehr einfach zu benutzen. Scratch legt den Schwerpunkt auf das Programmieren von grafischen Programmen wie zum Beispiel Spielen. Falls Sie sich noch nie mit irgendeiner Programmiersprache beschäftigt haben, ist Scratch ein prima Einstieg für Sie, und wir versuchen, Ihnen die Sprache auf den folgenden Seiten behutsam nahezubringen. Vor allem macht das Arbeiten mit Scratch Spaß, also nichts wie los!

Sofern Sie Raspbian als Betriebssystem auf Ihrem Pi haben, finden Sie das Scratch-Entwicklungswerkzeug bereits auf der Desktopumgebung, brauchen es also nicht gesondert zu installieren. Klicken Sie einfach auf das entsprechende Icon (Punkt 9 auf Seite 35). Falls Sie ein anderes Betriebssystem auf dem Pi benutzen und Scratch erst noch installieren müssen oder aber Scratch auf einem anderen Rechner, der mit Linux, Windows oder OS X läuft, ausprobieren möchten, konsultieren Sie bitte die offizielle Informationsseite **http://scratch.mit.edu**.

Das Hauptfenster der Scratch-Entwicklungsumgebung ist in mehrere Sektionen unterteilt. Grob gesagt sind die Programmier-Tools auf der linken Seite, das aktuell entstehende Programm in der Mitte und der Test- und Ausführbereich auf der rechten Seite



› **Sie können Ihre Scratch-Programme mit der Welt teilen, indem Sie sie auf scratch.mit.edu hochladen. Das geht ganz einfach direkt aus dem Menü der Entwicklungsumgebung heraus.**

Scratch

angeordnet. Jedes Scratch-Programm umfasst eine Anzahl grafischer Objekte (Sprites, Figuren) – etwa eine orangefarbene Cartoon-Katze –, die wiederum diverse Skripte enthalten. Diese Skripte bestimmen, was passiert, wenn das Programm läuft.

Oben links im Fenster der Entwicklungsumgebung sehen Sie acht Schaltflächen mit den Bezeichnungen **Bewegung**, **Aussehen**, **Klang**, **Malstift**, **Steuerung**, **Fühlen**, **Operatoren** und **Variablen**, die jeweils für eine Kategorie von Befehlen stehen, die Sie per Drag-and-Drop in die Skript-Abteilung im mittleren Bereich verschieben können, um so nach und nach Ihr Programm aufzubauen.

Wir beginnen mit einem sehr einfachen Programm. Klicken Sie auf den Button **Steuerung** oben links. Es erscheinen mehrere Schaltflächen, von denen vier an der Oberseite eine Wölbung aufweisen. Diese sind mit **Wenn <Flagge> angeklickt**, **Wenn Taste <Leertaste> gedrückt**, **Wenn Objekt1 angeklickt** und **Wenn ich <...> empfange** beschriftet. Die besondere Form der Schaltflächen weist darauf hin, dass es diese sind, mit denen ein neues Skript angelegt werden kann. Die restlichen Schaltflächen auf der linken Seite sind diesen vier untergeordnet.

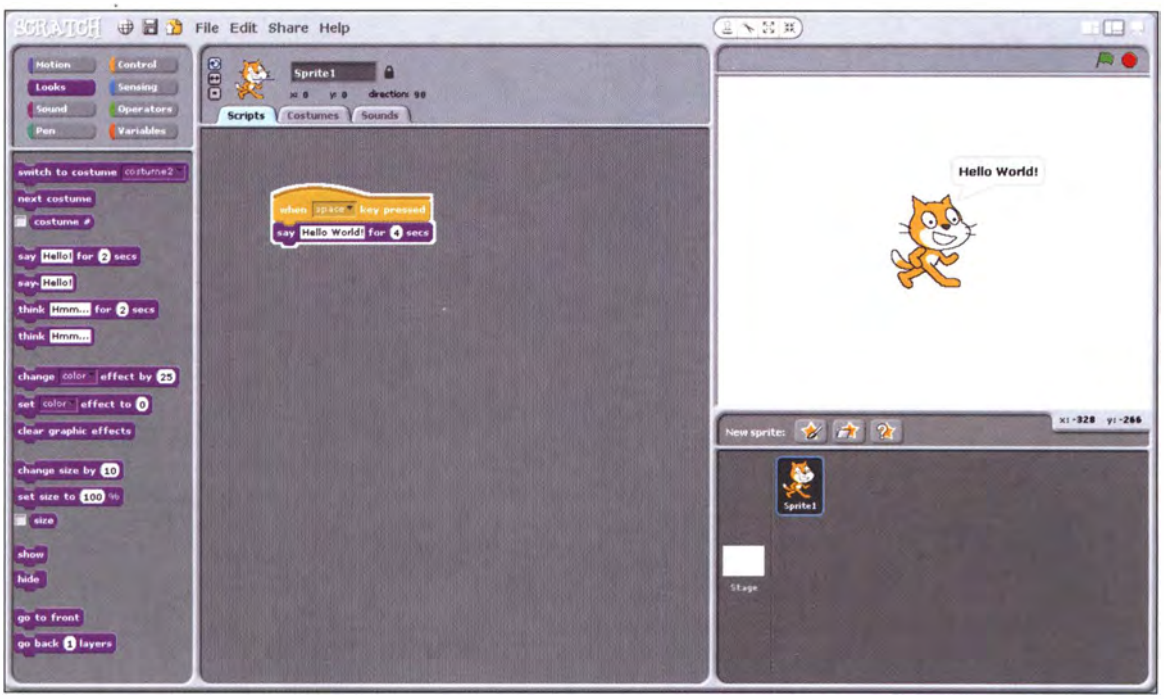
Verschieben Sie **Wenn Taste <Leertaste> gedrückt** per Drag-and-Drop in die Skript-Abteilung in der Mitte. Dort können Sie der Leertaste einen Vorgang zuordnen, der immer dann ausgeführt werden soll, wenn sie betätigt wird. Wir könnten zum Beispiel die Katze etwas sagen lassen. Klicken Sie dazu auf den Button **Aussehen** oben links, und ziehen Sie von den sich öffnenden Optionen **Sage <Hallo!> für <2> Sek.** in die Mitte. Die neue Anweisung sollte daraufhin an die bereits vorhandene andocken. Damit ist bereits das erste Skript fertig! Wenn Sie nun die Leertaste drücken, wird über unserer Katze 2 Sekunden lang eine Sprechblase mit dem vorgegebenen Text eingeblendet.

Sie können das Skript nun modifizieren, indem Sie den Text und die Anzeigezeit ändern. Klicken Sie dazu einfach in die entsprechende Stelle der Anweisung hinein, schreiben Sie etwas anderes dorthin, und bestätigen Sie mit der Eingabetaste. Nehmen wir an, Sie hätten „Hello World!“ als Sprechblasentext eingegeben, dann sähe unsere Katze – bei Betätigung der Leertaste – jetzt aus wie unten abgebildet.

Im nächsten Schritt wollen wir ein wenig Bewegung in das Programm hineinbringen. Klicken Sie erneut auf **Steuerung**, und ziehen Sie die Schaltfläche **Wenn Taste <Leertaste> gedrückt** ein zweites Mal in die Mitte. Ändern Sie die Tastenzuordnung – zum Beispiel in **Pfeil nach oben**. Holen Sie anschließend noch zwei weitere Kopien der Schaltfläche in die Mitte, sodass sie insgesamt vier davon haben, und ordnen Sie den letzten beiden die Tastenbelegungen **Pfeil nach unten** und **Pfeil nach rechts** zu.

Mithilfe der drei neuen Skripte werden wir das Objekt, also unsere Katze, bewegen können. Klicken Sie auf den Button **Bewegung** oben links, um sich die entsprechenden Optionen anzeigen zu lassen. Ziehen Sie die drei Schaltflächen **drehe dich <im Uhrzeigersinn> um <15> Grad** und **drehe dich <gegen den Uhrzeigersinn> um <15> Grad** sowie **gehe <10>er-Schritt** in den mittleren Bereich, und ordnen Sie diese – wie gehabt durch Andocken – den drei Pfeiltasten zu. Nun können Sie die Figur durch Drücken der Pfeiltasten bewegen. Fertig ist Ihr zweites Skript!

Das dritte Programm ist ein wenig komplexer: Darin kommt ein zweites Objekt – eine Maus – hinzu, die von der Katze gejagt wird und ihr ausweichen muss. Aber keine Angst, wir gehen die einzelnen Schritte nacheinander durch. Die Anleitung beginnt auf Seite 49.



› „Hello World!“ oder „Hallo Welt!“ gehört zum Standardvokabular jedes Programmierers. Seit Mitte der 1970er Jahre ist es eine typische Testphrase.

Das Interface der Scratch-Entwicklungsumgebung

Skripte
Hier werden die Skripte für das derzeit bearbeitete Objekt angezeigt.

Veröffentlichen
Sie können Ihre Scratch-Programme mit der Welt teilen, indem Sie sie auf **scratch.mit.edu** hochladen.

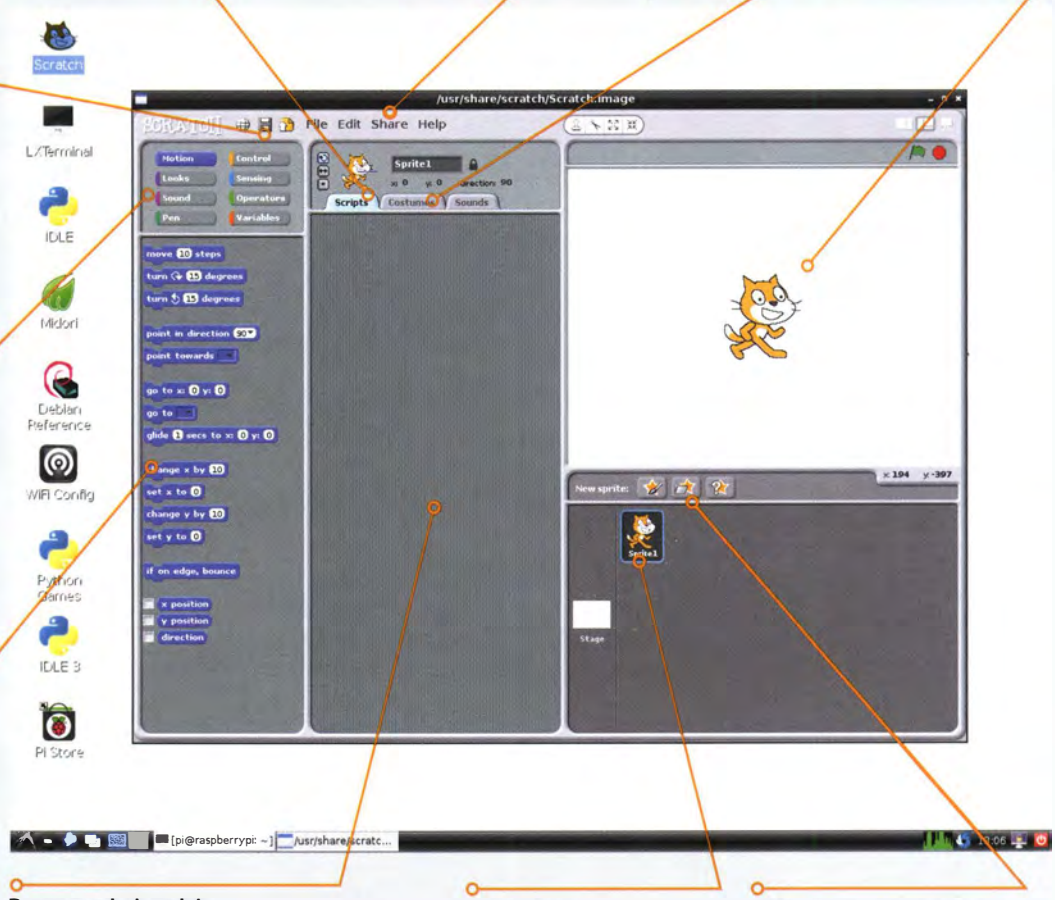
Kostüme
Mit Kostüme sind die verschiedenen bildlichen Umsetzungen der Objekte gemeint.

Bühne
Hier werden die Objekte dargestellt und bewegt.

Speichern
Hier können Sie Ihre Programmdatei sichern, was Sie auch regelmäßig tun sollten, damit Sie nicht ganz von vorn beginnen müssen, wenn etwas schief läuft.

Blockbereiche
Hier sind die Befehlskategorien **Bewegung**, **Aussehen**, **Klang**, **Malstift**, **Steuerung**, **Fühlen**, **Operatoren** und **Variablen** angeordnet. Jeder Blockbereich enthält ein eigenes Set von Befehlen, die alle mit derselben Farbe gekennzeichnet sind.

Code-Optionen
Hier sind die Befehlselemente aufgeführt, aus denen das Skript gebaut wird. Sie lassen sich per Drag-and-Drop in den mittleren Bereich ziehen.

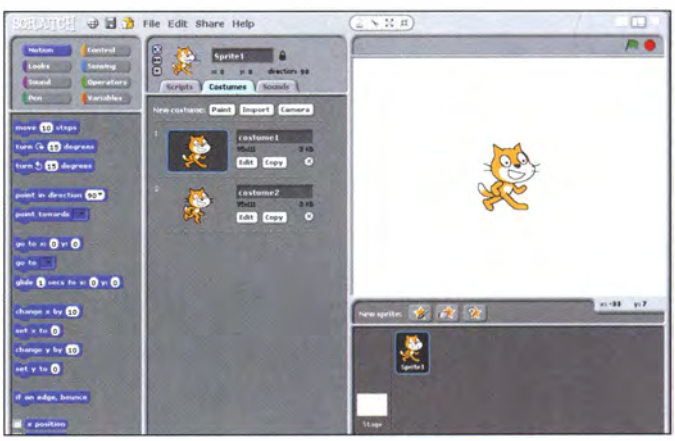


Programmierbereich
Im mittleren Teil des Fensters setzen Sie die Code-Elemente zu einem Skript zusammen. Hier werden auch die Kostüme gestaltet und Sounds bearbeitet. Zwischen den verschiedenen Funktionen können Sie mithilfe der drei Reiter umschalten.

Objektliste
Hier werden die Objekte (Sprites, Figuren) angezeigt, die in Ihrem Programm vorkommen.

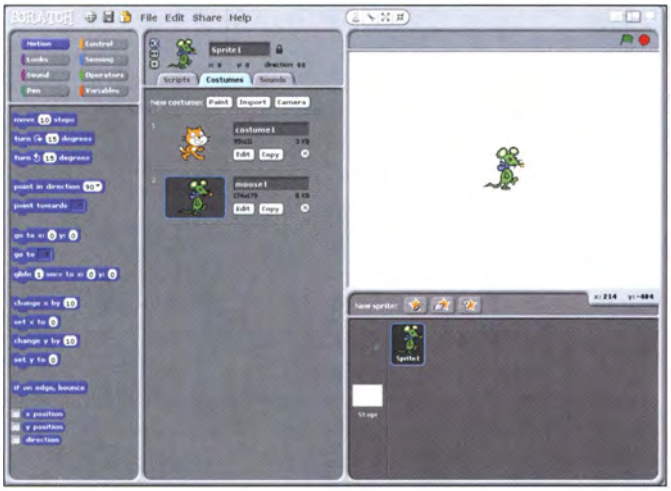
Objekt aus einer Datei laden
Klicken Sie hier, um ein neues Scratch-Objekt aus einer vorhandenen Bilddatei zu erzeugen. Mit dem Icon links daneben können Sie aber auch selbst ein neues Objekt zeichnen.

› **Scratch bietet Ihnen eine Reihe vorgefertigter Objekte an, aus denen Sie wählen können. Sie können den Katalog über die Objektliste oder den Kostüme-Reiter im Programmierbereich aufrufen.**



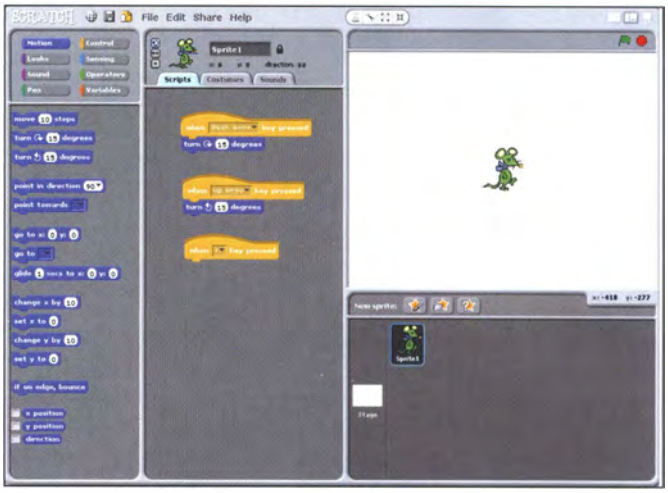
› **Jedes Objekt kann mehrere Kostüme (bildliche Umsetzungen) haben. Im Programmierbereich werden diese angezeigt.**

Schritt für Schritt



1 Die Maus kreieren

Beim Programmieren des Spiels beginnen wir mit der Maus. Wir geben dazu einfach unserem **Objekt1** ein neues Erscheinungsbild. Gehen Sie im Menü über **Kostüme** > **Importieren** > **Tiere** zu dem gewünschten Bild der Maus und wählen Sie es aus. Mithilfe des Schrumpfen-Icons oberhalb der Bühne können Sie die Maus noch verkleinern.



2 Tastenzuordnung

Gehen Sie nun im Programmierbereich auf den Reiter **Skripte**, und ersetzen Sie bei der Schaltfläche **Wenn Taste <Pfeil nach rechts> gedrückt** die Tastenzuordnung durch den Buchstaben **r** wie Reset. Damit wird später ein neues Spiel gestartet. Die Anweisung **gehe <10>er-Schritt** entfernen Sie, indem Sie sie aus dem Programmierbereich ziehen.

3 Variablen erstellen und benennen

Klicken Sie auf den Button **Variablen** bei den Blockbereichen oben links. Erstellen Sie zwei Variablen mit den Namen **score** (Punktestand) und **over** (Ende).

4 Den Punktestand zurücksetzen

Setzen Sie die Schaltflächen **zeige dich** (aus dem Blockbereich **Aussehen**), **gehe zu x: <100> y: <100>** (aus **Bewegung**) sowie **setze <score> auf <0>** und **setze <over> auf <0>** (beide aus **Variablen**) unter das Skript **Wenn Taste <r> gedrückt**. Wie Sie gesehen haben, müssen Sie die Werte für **x** und **y** zuerst anpassen.

Variablen und Messages

Früher oder später werden Sie sich wünschen, dass Ihr Programm sich Dinge merkt, etwa eine Zahl oder ein Stück Text. Dies können Sie mithilfe von Variablen erreichen. Variablen sind kleine Bereiche im Speicher Ihres Computers, wo Ihr Programm Daten ablegen kann. In Schritt 3 der Anleitung oben haben wir bereits zwei Variablen erstellt. Bei anderen Programmiersprachen können Sie noch ganz andere Arten von Variablen erstellen, doch das ist an dieser Stelle noch nicht relevant. Sobald Sie eine Variable erstellt haben, können Sie Verschiedenes damit tun: Sie können sie auf einen bestimmten Wert setzen, die Erfüllung einer Bedingung überprüfen oder sie ausgeben lassen.

Messages

Wenn Sie mehrere Skripte gebaut haben, möchten Sie vielleicht, dass diese miteinander kommunizieren. Dies lässt sich manchmal mittels Variablen bewerkstelligen, doch die sogenannten Messages sind häufig besser geeignet. Messages können Skripte starten, so wie etwa die Betätigung einer bestimmten Taste einen bestimmten Vorgang auslösen kann. Wenn ein Skript eine bestimmte Message versendet, ruft es damit alle anderen Skripte auf, die mit **Wenn ich <Message> empfange** beginnen.

5 Sendebefehl hinzufügen

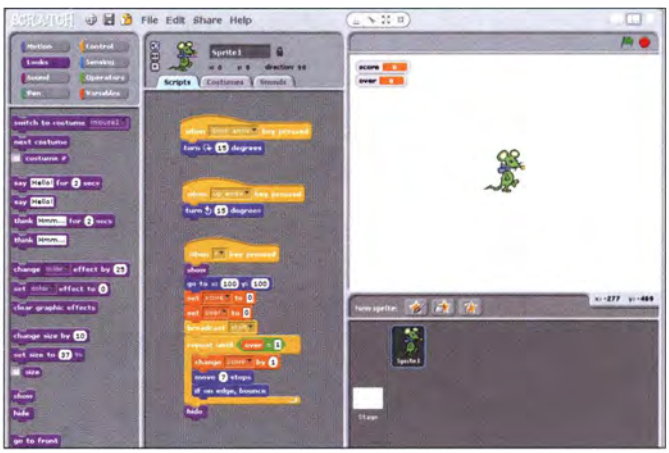
Fügen Sie die dem Skript **Wenn Taste <r> gedrückt** noch die Schaltfläche **sende <...>** hinzu, und erstellen Sie im Aufklappmenü dieser Schaltfläche eine neue Message mit dem Namen **start**.

6 Eine Schleife erstellen

Sie können Schleifen erstellen, die denselben Code mehrfach ausführen, bis eine Bedingung erfüllt ist. Erweitern Sie das Skript mit **wiederhole bis <...>** (aus **Steuerung**), und holen Sie sich eine Gleichung **<...> = <...>** (aus **Operatoren**) dazu. Ziehen Sie dann die Variable **over** in das Feld links vom Gleichheitszeichen und tippen Sie in das rechte Feld eine **1**.

7 Die Schleife erweitern

Zu dem Block **wiederhole bis <over> = 1** kommen jetzt noch die Anweisungen **ändere <score> um <1>** (aus **Variablen**), **gehe <7>er-Schritt** (aus **Bewegung**) und **pralle vom Rand ab** (ebenfalls aus **Bewegung**). Diese drei Anweisungen werden so lange wiederholt, bis **over** auf **1** steht.



8 Die Maus verstecken

Wenn die Katze die Maus gefangen hat, endet die **wiederhole**-Schleife und das Programm schreitet weiter. Ziehen Sie die Anweisung **verstecke dich** (aus **Aussehen**) unter die Schleife, dann verschwindet die Maus, sobald sie gefangen wurde.

9 Die Katze kreieren

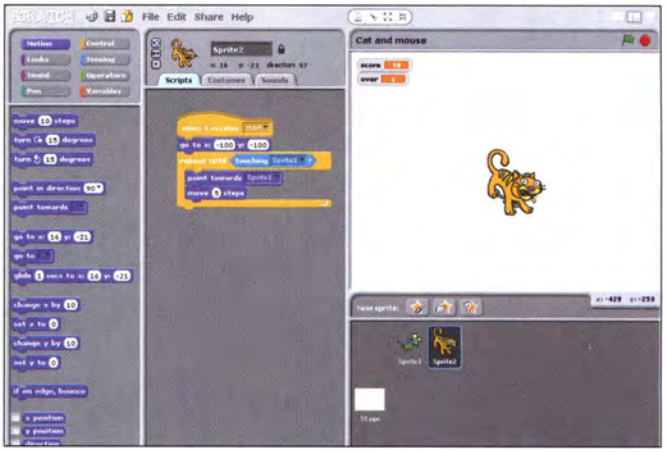
Da das **Objekt1** zur Maus mutiert ist, brauchen wir ein neues **Objekt2**, um die Katze wieder ins Spiel zu bringen. Erstellen Sie das Objekt, und gehen Sie bei der Auswahl des Erscheinungsbildes vor wie in Schritt 1. Auch die Katze sollte mithilfe des Schrumpfen-Icons verkleinert werden.

10 Die Katze bewegen

Starten Sie das neue Skript für die Katze mit der Anweisung **Wenn ich <start> empfange** (aus **Steuerung**), wobei **start** die Message ist, die wir in Schritt 5 erstellt haben. Fügen Sie dann **gehe zu x: <-100> y: <-100>** (aus **Bewegung**) hinzu. Dies bewegt die Katze in die Ecke gegenüber der Maus (die Mittelposition ist **x: <0> y: <0>**).

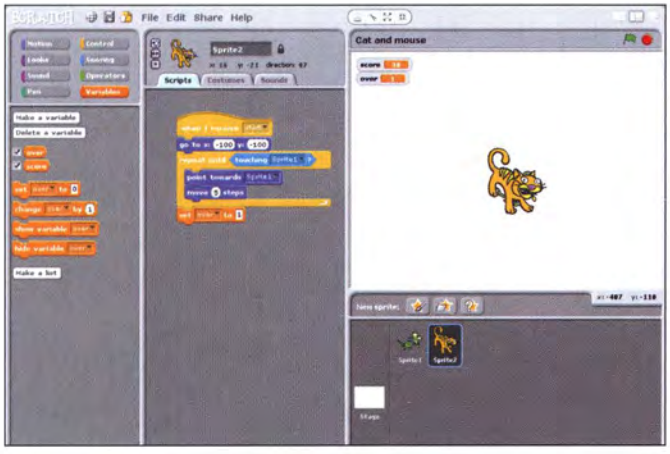
11 Eine neue Schleife erstellen

Auch die Katze benötigt eine Schleife, damit das Spiel am Laufen gehalten wird. Fügen Sie **wiederhole bis <...>** (aus **Steuerung**) hinzu, und setzen Sie **wird <Objekt1> berührt?** (aus **Fühlen**) in das leere Feld der Anweisung. Die Schleife endet also, sobald die Katze die Maus gefangen hat.



12 Die neue Schleife erweitern

Doch was genau soll die Katze so lange tun, bis sie die Maus gefangen hat? Dies bestimmen die Anweisungen **drehe dich zu <Objekt1>** und **gehe <4>er-Schritt** (beide aus **Bewegung**), die Sie an den **wiederhole**-Block anhängen. Aus dem Verhältnis der Schrittzahlen von Katze und Maus (in unserem Beispiel **4** und **7**) ergibt sich der Schwierigkeitsgrad des Spiels.

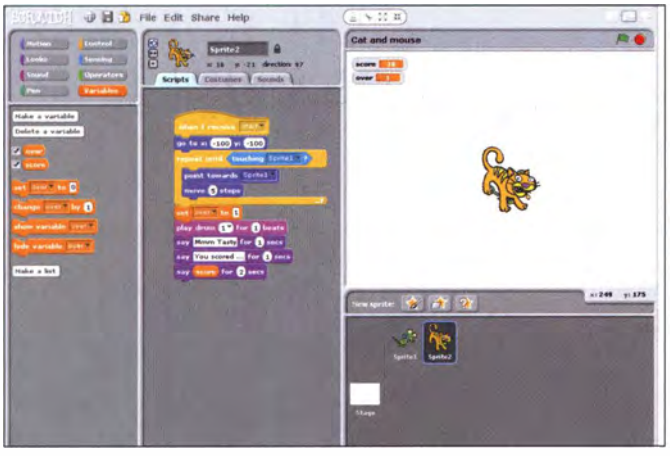


13 Beide Schleifen beenden

Die Schleife der Katze endet, wenn sie die Maus erreicht. Damit in diesem Moment auch die Schleife der Maus endet, muss die Bedingung aus Schritt 7 erfüllt werden, nämlich, dass **over** auf **1** steht. Darum fügen wir dem **wiederhole**-Block der Katze die Anweisung **setze <over> auf <1>** (aus **Variablen**) hinzu.

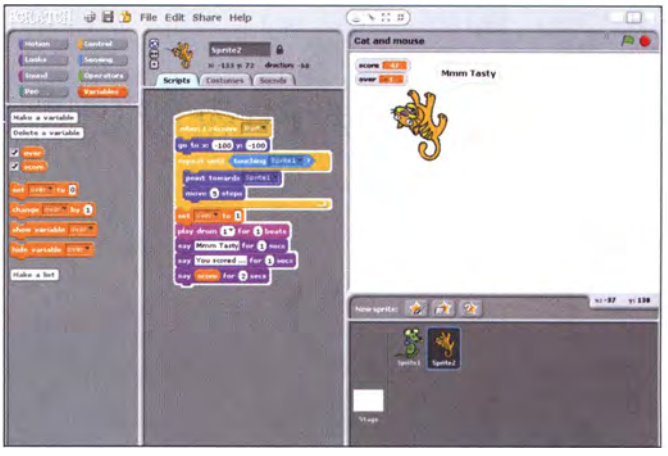
14 Spiel beendet

An diesem Punkt soll dem Spieler mitgeteilt werden, dass das Spiel vorbei ist. Um das zu tun, fügen wir dem Skript der Katze die Anweisung **Sage <Hab' ich dich!> für <2> Sek.** (aus **Aussehen**) hinzu.



15 Punktestand einblenden

Zuletzt soll dem Spieler noch angezeigt werden, wie viele Punkte er erzielt hat. Die Variable **score** in der Schleife der Maus wurde bei jedem Durchgang um **1** erhöht, und der Endwert soll uns als Punktestand dienen. Um ihn anzeigen zu lassen, müssen Sie noch **Sage <Erzielte Punkte:> für <1> Sek.** und **Sage <score> für <5> Sek.** anhängen.



16 Achtung, fertig, los!

Drücken Sie die Taste **r** und retten Sie die Maus vor der Katze! Da die Anweisungen **drehe dich <im Uhrzeigersinn> um <15> Grad** und **drehe dich <gegen den Uhrzeigersinn> um <15> Grad** aus Ihrem zweiten Skript noch den Pfeiltasten zugeordnet sind, können Sie die Maus mit den Pfeilen steuern. Viel Spaß und toi, toi, toi! ■

Netzwerk: SSH-

Nutzen Sie öffentliche WLAN-Netzwerke ohne Risiko und umgehen Sie Online-Zensur mithilfe Ihres Raspberry Pi. Jonathan Roberts führt Sie in die Technik des SSH-Tunneling ein.

Die meisten von uns hatten bestimmt schon mal mit dem Problem einer eingeschränkten Internetverbindung zu tun, sei es aufgrund von Firewalls, die die Nutzung von Diensten wie SMTP-E-Mail oder XMPP blockieren, oder von Web-Filtern, die bestimmte Seiten oder Inhalte ausschließen. Das kann mitunter ganz schön nervig sein.

Viele von uns kennen vermutlich auch ein ungutes Gefühl bei der Nutzung öffentlicher, kostenloser WLAN-Verbindungen, da es für andere Nutzer desselben Netzwerks theoretisch recht einfach ist, unsere Aktivitäten darin auszuspähen oder gar zu manipulieren. In einem freien Netzwerk weiß man nie, mit wem man es teilt, und ebenso wenig weiß man, was der Betreiber des WLAN-Hotspots mit dem Datentraffic anstellt.

In diesem Artikel zeigen wir Ihnen, wie Sie bei Ihrem Raspberry Pi mithilfe eines SSH-Tunnels eine verschlüsselte Netzwerkverbindung einrichten und dadurch Zensoren und Spione in die Schranken weisen. Der Raspberry Pi ist dafür das perfekte Gerät: Klein und mit niedrigen Unterhaltskosten, können Sie ihn einfach in einem Schrank verstauen, bis Sie ihn benötigen.

Raspbian

Bevor wir zum eigentlichen Thema kommen, schnell noch ein paar Worte zum Betriebssystem Ihres Raspberry Pi. Falls Sie zu den Pi-Fans der ersten Stunde gehören, verwenden Sie vielleicht noch Debian. Inzwischen empfiehlt die Raspberry-Pi-Stiftung stattdessen das für die Hardware des Geräts optimierte Raspbian, das eine Weiterentwicklung von Debian darstellt.

> „Raspbian, eine Weiterentwicklung von Debian, wartet mit vielen spezifischen Verbesserungen auf."

Raspbian wartet mit vielen spezifischen Verbesserungen auf: Nicht nur funktioniert der Ton sofort nach dem Einschalten, auch die Arbeitsgeschwindigkeit des Pi hat sich wesentlich erhöht (einige Benchmarks zeigen Geschwindigkeitszuwächse von 4-40 % in Abhängigkeit von der jeweils verwendeten Funktion). Zusätzlich bringt Raspbian ein automatisiertes Einrichtungstool mit, das beim ersten Einschalten des Raspberry Pi startet.

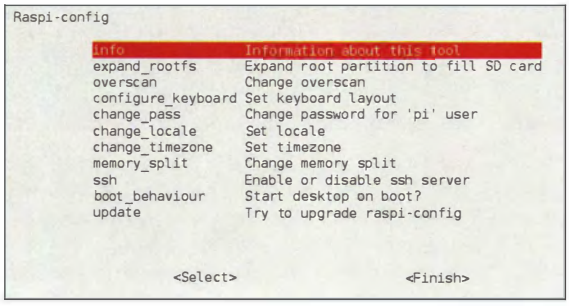
Dieses macht die meisten Anpassungen, die zu Beginn der Arbeit am Raspberry Pi notwendig sind, sehr einfach. Dazu gehört neben der Anpassung der Größe der Root-Partition (die sich nun sehr einfach auf die gesamte Kapazität der SD-Karte ausdehnen lässt), dem Ausschalten von Overscan und der Änderung des Tastaturlayouts auch die Aktivierung von SSH.

Auch wir empfehlen deshalb, dass Sie auf die neueste Version von Raspberry umsteigen. Der einfachste Weg ist es, das aktuellste Image von **www.raspberrypi.org/downloads** herunterzuladen, es zu entpacken und mit **dd** auf die SD-Karte kopieren.

Sie sollten möglichst auch alle Inhalte von der SD-Karte löschen, da andernfalls Probleme entstehen können, falls alte Programme und Konfigurationsdateien mit der neuen Software in Konflikt geraten. Dies geht mit folgendem Kommando:

```
sudo dd if=/dev/zero of=/dev/mmcblk0 bs=1M
```

Ersetzen Sie **mmcblk0** dabei mit dem Gerät, das Ihre SD-Karte repräsentiert. Vergessen Sie nicht, vorher alle wichtigen Daten auf der SD-Karte zu sichern. Dieser Befehl überschreibt nämlich das komplette Laufwerk mit Nullen, sodass es danach unmöglich ist, die alten Daten wiederherzustellen.



Das raspi-config-Werkzeug macht es sehr einfach, Ihren Raspberry Pi zu konfigurieren. Um es aufzurufen, müssen Sie nur den Befehl sudo raspi-config eingeben.

Was sind Ports?

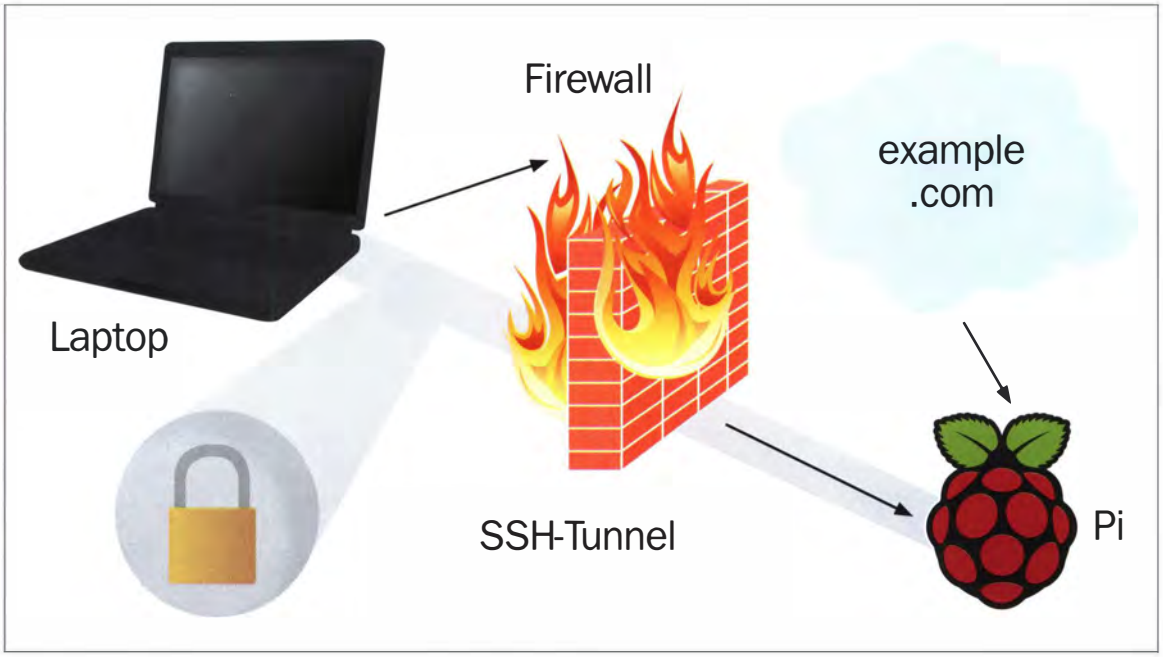
SSH-Tunneling ist eine Möglichkeit, eine geschützte Netzwerkverbindung aufzubauen. Vereinfacht gesagt handelt es sich um das clevere Weiterleiten von Ports. Wenn zwischen zwei Computern eine Verbindung hergestellt werden soll – ob nun Dokumente transferiert, Secure Shells (SSH) geöffnet oder Dateien per NFS ausgetauscht werden sollen –, müssen dem Computer, der die Verbindung einleitet, zwei Informationen bekannt sein: die IP-Adresse des Zielrechners sowie die Portnummer des Dienstes (beispielsweise HTTP, SSH oder NFS).

Aber was ist ein Port? Vermutlich kennen Sie das Konzept der IP-Adresse. Dies ist eine Nummer (beispielsweise 192.168.133.20), die Computer in einem Netzwerk eindeutig identifiziert, in etwa so, wie Ihre Postadresse Ihr Haus adressiert.

Genauso, wie in einem Haus meistens mehrere Personen leben, die alle auf Post warten, gibt es nun auf einem PC viele verschiedene Dienste, die auf an sie gerichtete Netzwerkverbindungen warten. In der realen Welt suchen wir uns den richtigen Empfänger aus, indem wir dessen Namen auf den Briefumschlag schreiben. Beim Computer geschieht das, indem wir die spezifische Portnummer des Dienstes mitsenden, mit dem wir uns verbinden möchten.

Jeder Port wird mit einer Zahl zwischen 0 und 65535 bezeichnet, und die auf dem Computer installierten Dienste hören einen oder mehrere dieser Ports auf eingehende Verbindungsanfragen ab. Die Internet Assigned Numbers Authority (IANA) führt eine Liste, die alle gängigen Dienste mit den zugehörigen Portadressen, auf denen sie normalerweise mithören,

Tunneling



› **Ein SSH-Tunnel erstellt eine sichere Verbindung zwischen zwei Computern, die Sie verwenden können, um Proxys und andere Filter zu umgehen.**

verzeichnet. Konkret werden die Ports 0 bis 1023 abgedeckt.

Ports können jedoch auch umdefiniert werden. Somit können Sie einen Dienst so konfigurieren, dass er einen anderen Port abhört als üblich. Solange Sie wissen, welchen Port der Dienst auf dem Server abhört, können Sie einen Client ohne Probleme anweisen, sich über diesen Port mit dem Server zu verbinden, anstatt dafür den Standardport zu verwenden.

Ihre Verbindung tunneln

Lassen Sie uns das SSH-Tunneling nun anhand eines praktischen Beispiels durchspielen. Angenommen, Sie betreiben zwei Geräte: eins mit der Kennung **laptop** – ein Notebook, das sich hinter einer Firewall befindet und beispielsweise in einem Internetcafé eine bestimmte gesperrte Website **www.example.com** nicht aufrufen kann – und eins mit der Kennung **pi** – das ist ihr Raspberry Pi, der zu Hause mit einem unbeschränkten Internetanschluss aufwarten kann. **laptop** kann außerdem auf **pi** zugreifen.

Damit Sie jetzt vom Rechner **laptop** aus trotzdem die Seite **www.example.com** erreichen können, müssen sie einen Tunnel einrichten. Konkret bedeutet das, dass Sie ihre Internetverbindung durch **pi** schleifen müssen. SSH-Tunneling ermöglicht Ihnen genau das.

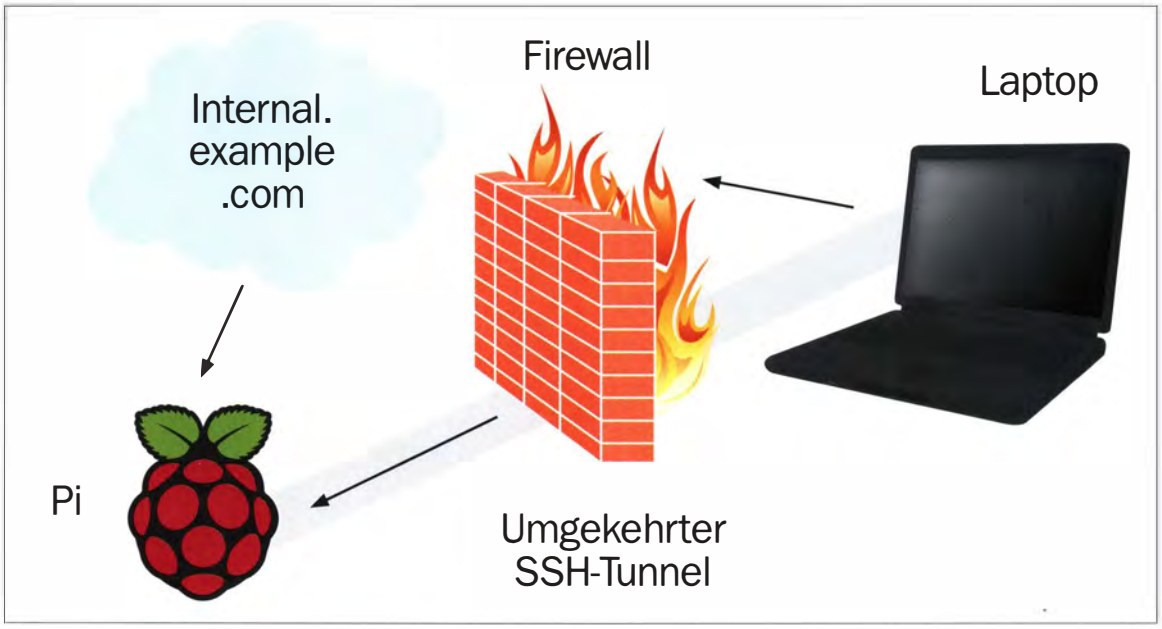
Die gewöhnliche Funktionsweise von SSH besteht darin, eine verschlüsselte Verbindung zwischen zwei Computern über den Port 22 herzustellen. Anstatt nur eine Shell-Verbindung zum entfernten Computer zu öffnen, nutzen Sie SSH in diesem Fall aber dazu, einen Tunnel einzurichten, der den kompletten Netzwerkverkehr eines angegebenen Ports von **laptop** auf Port 22 umleitet und ihn so verschlüsselt an **pi** sendet.

Dadurch haben sich gleich zwei Vorteile. Erstens: Der Datenverkehr wird – wie auch eine normale SSH-Verbindung – verschlüsselt, sodass niemand überwachen kann, was Sie über diese Verbindung senden, und somit beispielsweise keine Benutzernamen und Passwörter abgehört werden können. Das Tolle am SSH-Tunneling ist, dass Sie das mit jeder Art von Netzwerkkommunikation machen können, nicht nur mit Login-Abfragen. Zweitens: Sie können Firewalls oder andere Web-Filter umgehen. Wenn der Router Ihrer lokalen Internetverbindung den Zugriff auf **www.example.com** verhindert, verschaffen Sie sich dennoch Zugang, indem Sie die Internetverbindung über einen ungefilterten Port an ein Gerät weiterleiten, dessen Internetverbindung nicht gefiltert wird. Damit das klappt, müssen Sie sicherstellen, dass SSH sowohl auf **laptop** als auch auf **pi** installiert ist und als Dienst läuft. Auf dem Raspberry Pi ist das sehr einfach über *raspi-config* zu bewerkstelligen, indem Sie die SSH-Option auswählen (vorausgesetzt, Sie verwenden die neueste Version von Raspbian).

Bei dem Notebook aus unserem Beispiel hängt es davon ab, welche Distribution Sie verwenden. Um SSH unter Fedora zu »

Was ist SSH?

Wenn Sie den Raspberry Pi bereits verwendet haben, wissen Sie, was die Kommandozeile ist. Wenn Sie das erste Mal Ihren Raspberry Pi starten, finden Sie einen schwarz-weißen Bildschirm vor, der von Ihnen die Eingabe Ihres Nutzernamens und des Passworts verlangt. Anschließend steht es Ihnen frei, noch weitere Befehle einzugeben, etwa zum Starten einer grafischen Benutzerumgebung, zum Bearbeiten einer Textdatei oder zum Verwenden eines Browsers wie *ELinks*. Eine andere Bezeichnung für die Kommandozeile ist „Shell“, und SSH seht für „Secure Shell“. SSH bietet eine sichere Möglichkeit, sich über ein fremdes Netzwerk in den eigenen Computer einzuloggen. Die Verbindung wird dabei mittels mächtiger Mathematik verschlüsselt. So ist niemand in der Lage, die Daten zu lesen, die Sie über das Netzwerk an Ihren Computer versenden, egal, ob es sich dabei um Ihre Benutzerdaten und Passwörter oder um Dateien, Internetadressen und Cookies handelt. Im Prinzip ist SSH eine moderne Alternative zu anderen Methoden des Fern-Logins wie Telnet oder rlogin.



Ein umgekehrter SSH-Tunnel ist nützlich, um Zugang zu geschützten Netzwerken zu bekommen. Sprechen Sie sich vor dessen Einrichtung aber unbedingt mit Ihrer IT-Abteilung ab, da Sie ansonsten Ärger bekommen könnten.

konfigurieren, sind folgende Schritte notwendig:

```
su -c "yum -y install openssh"
su -c "systemctl enable sshd.service"
su -c "systemctl start sshd.service"
```

Unter Ubuntu oder Debian erledigen Sie das Gleiche mit:

```
sudo apt-get install openssh-server openssh-client
sudo service ssh start
sudo insserv ssh
```

Falls auf Ihrem Notebook Windows läuft, sollten Sie sich *Putty* oder *Cygwin* ansehen, auf die wir hier jedoch nicht weiter eingehen.

Den Tunnel anlegen

Sobald Sie die Einrichtung von SSH erledigt haben, können Sie nun das für die Erstellung des SSH-Tunnels notwendige Kommando ausführen:

```
ssh -L 1080:example.com:80 pi-user@pi -f
```

Wir erklären diesen Befehl im Einzelnen, damit damit Sie genau verstehen, was hier passiert. Der Schalter **-L** teilt SSH mit, dass wir es benutzen werden, um damit einen Port weiterzuleiten.

Das anschließende Textstück wird von Doppelpunkten in drei Segmente unterteilt. Die erste Zahl gibt den Port auf dem lokalen Rechner – in unserem Fall **laptop** – an, der weitergeleitet werden soll. Jeder Port über 1024 kann ohne Root-Rechte nach Belieben weitergeleitet werden. Alle Daten, die an diesen Port

> **„Um die Firewall von außen zu umgehen, können Sie einen umgekehrten Tunnel erstellen."**

geschickt werden, werden versendet. Das mittlere Segment gibt den Rechner an, an den die Daten weitergeleitet werden sollen (dieser kann statt mit dem Domainnamen auch in der Form einer IP-Adresse angegeben werden). Das letzte Segment bezeichnet den Port des Zielrechners, an den das Ganze gehen soll. Dieser Wert kann beliebig gewählt werden, solange der User **pi** die Berechtigung hat, diesen Port zu verwenden. Da wir auf eine Internetseite zugreifen möchten, geben wir hier den Port 80 an – das ist der Standardport für Webtraffic.

Zum Schluss geben wir den Benutzernamen und die Bezeichnung des Rechners an, durch den wir die Verbindung hindurchleiten wollen. In diesem Fall ist das der Nutzer **pi-user** auf dem Computer **pi**.

Der **-f**-Schalter teilt SSH mit, dass es im Hintergrund laufen und auf **pi** nicht das Terminal blockieren soll.

Um nun vom Rechner **laptop** aus auf **www.example.com** zugreifen zu können, müssen Sie nur noch mit dem Browser die Seite **http://localhost:1080** aufrufen. Nun sollte alles funktionieren. Indem Sie jeder Webadresse ein **:1080** anhängen, wird der Browser angewiesen, den Port 1080 statt des standardmäßig festgelegten Ports 80 zu verwenden. Das **localhost** veranlasst den Browser, auf den eigenen Rechner über Port 1080 zuzugreifen: Es ist sozusagen die Internetadresse des lokalen Rechners.

Sie müssen die Vorgehensweise nur leicht modifizieren, dann können Sie das Gleiche auch mit anderen Diensten als WWW durchführen. Falls Sie Ihren eigenen Mailserver (SMTP) auf dem Raspberry Pi hosten, könnten Sie bei den Einstellungen Ihres Mailprogramms **www.example.com** zu localhost und Port 80 zu 25 ändern, und alles wäre eingestellt.

Umgekehrte Tunnel

SSH ist auch in der Lage, andere Arten von Tunneln anzulegen. Lassen Sie uns diese kurz ansehen. Im vorherigen Beispiel haben wir einen lokalen Port weitergeleitet (durch den Schalter **-L** angezeigt).

Stellen wir uns nun ein leicht abgewandeltes Szenario vor. Wieder haben wir zwei Computer mit den Kennungen **laptop** und **pi**. Sie haben das Notebook zu Hause, aber Sie benötigen Zugang zu einer internen Webseite wie **internal.example.com**. Diese befindet sich auf einem Netzwerk, das für Außenstehende durch eine Firewall versperrt ist. Der Rechner **pi** ist hingegen mit dem internen Netzwerk verbunden.

Um die Firewall von außen zu umgehen, können Sie einen umgekehrten Tunnel erstellen, der **laptop** und **pi** verbindet. Das Notebook kann diesen Tunnel benutzen, um über den Raspberry Pi in das durch die Firewall geschützte Netzwerk zu gelangen und Zugriff auf **internal.example.com** zu bekommen.

Dazu müssen Sie auf dem **Raspberry Pi** folgendes Kommando ausführen:

```
ssh -R 1080:internal.example.com:80 laptop-user@laptop -f
```

Das -R steht für einen entfernten oder umgekehrten Tunnel. Im nächsten Teil des Befehls wird der Port auf dem Notebook

angegeben, von dem die Verbindung weitergeleitet werden soll.

Der Rest der Angaben funktioniert genauso wie im vorherigen Beispiel. Um diese Technik optimal nutzen zu können, müssen Sie den SSH-Tunnel im Voraus anlegen, da Sie auf den Raspberry Pi mit dem Notebook nicht zugreifen können. Außerdem muss die SSH-Verbindung des Raspberry Pi ständig aktiv bleiben. Das erreichen Sie, indem Sie **autossh** anstatt dem normalen **ssh** verwenden. Allerdings weicht der Befehl unter **autossh** leicht ab:

```
autossh -M 20000 -f 1080:internal.example.com:80
laptopuser@laptop
```

Achten Sie auf den Schalter **-M**. Hier müssen Sie einen Port angeben, der ansonsten komplett unbenutzt ist. Wählen Sie also eine hohe Portnummer. **autossh** nutzt ihn, um die SSH-Verbindung zu überwachen und wiederherzustellen, falls Sie beendet wird.

Der Trick des umgekehrten Tunnelns ist sehr nützlich, könnte Ihnen allerdings Ärger mit dem jeweiligen Systemadministrator einbringen, sofern dieser davon Wind bekommt. Es gibt nämlich gute Gründe für die Errichtung von Firewalls, und Leute, die Löcher hineinreißen, sind bei Administratoren sehr unbeliebt.

Einige Sicherheitsexperten nutzen diese Technik, um Firmennetzwerke auf ihre Sicherheit hin zu überprüfen. Sie schmuggeln dazu kleine „Drop-Boxes“ (wofür der Raspberry Pi wiederum perfekt geeignet wäre) in die Firma und verbinden diese unauffällig mit irgendeinem Netzwerkstecker. Die Geräte sind so konfiguriert, dass sie automatisch einen umgekehrten SSH-Tunnel (wahrscheinlich über **/etc/network/if-up.d**) anlegen, über den die Sicherheitsexperten dann in das Firmennetzwerk eindringen und Schaden anrichten können.

Dynamische Tunnel

Es gibt noch eine weitere Tunnel-Methode für SSH, die wahrscheinlich die nützlichste ist. Anstatt die Ports oder Dienste explizit anzugeben, um Daten über sie versenden zu können, kann man auch dynamische Tunnel benutzen. Diese ermöglichen es, die Daten aller Dienste zu sammeln und zu versenden. Auch der dazu nötige Befehl ist simpler:

```
ssh -D 1080 pi-user@pi
```

Der Schalter **-D** teilt SSH mit, dass wir einen dynamischen SSH-Tunnel einrichten möchten. Die Portnummer gibt den lokalen Port an, während **pi-user@pi** den Namen des Nutzers und des Rechners angibt, zu denen der Tunnel führen soll.

Nachdem Sie diesen Befehl ausgeführt haben, können Sie jede Anwendung auf Ihrem lokalen Rechner so konfigurieren, dass Ihr Netzwerkverkehr durch den SSH-Tunnel geleitet wird, den Sie gerade angelegt haben. Voraussetzung ist allerdings, dass die Anwendung SOCKS-Proxys unterstützt.

Unter *Firefox* erreichen Sie das beispielsweise, indem Sie Extras > Einstellungen > Erweitert > Netzwerk > Verbindung > Einstellungen aufrufen und dort die manuelle Proxy-Konfiguration auswählen. Geben Sie dort 127.0.0.1 als SOCKS-Host ein, dazu die Port-Nummer, die Sie im SSH-Befehl verwendet haben.

Nach einem Klick auf OK sollte alles fertig eingerichtet sein. Nun sollten Sie mit *Firefox* ganz normal surfen können. Der Netzwerkverkehr des Browsers wird dabei allerdings vollständig durch den SSH-Tunnel zum Raspberry Pi geleitet.

Sie können das überprüfen, indem Sie die SSH-Verbindung beenden (mit STRG + C) und anschließend nochmals versuchen, eine Website aufzurufen. Falls der Tunnel funktioniert, sollten Sie jetzt nicht mehr in der Lage sein, irgendeine Internetseite aufzurufen.

> **„Es gibt noch eine weitere Tunnel-Methode für SSH, die wahrscheinlich die nützlichste ist.“**

Andere mit dem Internet verbundene Anwendungen wie E-Mail-Clients und Instant-Messenger haben ähnliche Konfigurationsbildschirme wie *Firefox*. Sie brauchen nur die Informationen des SOCKS-Proxys anzupassen, um diese über den Tunnel laufen zu lassen.

Die hier vorgestellten Techniken sind sehr praktisch, wenn Sie unterwegs eine sichere und unbeschränkte Netzwerkverbindung benötigen. Allerdings funktioniert all das nur, wenn der als Tunnelserver dienende Rechner permanent läuft. Hier zeigt sich eine weitere Stärke des Raspberry Pi, denn sein Stromverbrauch ist sehr gering. ■

Dynamisches DNS

Die meisten der hier vorgestellten Techniken verlangen die Verwendung einer öffentlich zugänglichen IP-Adresse oder Domain. Bei heimischen Internetanschlüssen erhält man in aller Regel eine solche öffentliche IP-Adresse zugeteilt, allerdings ist sie zumeist dynamisch, das bedeutet, sie verändert sich in bestimmten Intervallen. Daher kann es vorkommen, dass Sie einen Tunnel konfigurieren, damit zu arbeiten beginnen und er plötzlich nicht mehr funktioniert, weil Ihre IP-Adresse sich geändert hat. Es gibt mehrere Möglichkeiten, dieses Problem zu lösen. Die einfachste ist es, einen Internetprovider zu wählen, der statische IP-Adressen vergibt. Einige Provider stellen diese auf Anfrage zur Verfügung, manche verlangen dafür eine Gebühr. Manche Anbieter vergeben jedoch gar keine statischen IP-Adressen. Falls das Wechseln des Internetproviders für Sie keine Option ist, können Sie auf einen Service wie **www.no-ip.com** oder **www.dyn.com** zurückgreifen. Für diese Dienste installieren Sie ein Programm auf dem Rechner, das Ihre IP-Adresse überwacht. Wenn Sie sich ändert, wird das von den Tools registriert und an den entsprechenden Dienstanbieter gemeldet. Dieser verwaltet eine Domain wie zum Beispiel **myip.dyn.com** und stellt sicher, dass diese immer auf Ihre aktuelle IP-Adresse verweist, egal, wie oft sich diese ändert. Falls sich Ihr Raspberry Pi hinter einem Router befindet, stellen Sie den Router so ein, dass der Port 22 zum Pi weitergeleitet wird. Diese Konfiguration nennt sich „Port Forwarding“ und erfolgt bei jedem Router etwas unterschiedlich. Normalerweise gibt es eine Web-Adresse, über die Sie die Konfigurationsoberfläche des Routers aufrufen. Von dieser Oberfläche aus können Sie bestimmte Ports an bestimmte Computer weiterleiten, die durch ihre MAC-Adresse identifiziert werden. Sie können dem Raspberry Pi auch eine statische IP zuweisen und diese als Zielangabe für den Port verwenden.

› no-ip.com erstellt einen dynamischen DNS-Eintrag, sodass Sie Ihren Raspberry Pi auch dann erreichen, wenn seine IP-Adresse wechselt.

Server: Chatten

Alex Cox errichtet ein IRC-Netzwerk von Grund auf und haucht ihm ein verrücktes Leben ein, und das alles mit nur 224 MB RAM.

Der Raspberry Pi – Lerninstrument oder Spielzeug? Man muss hier gar keine Entscheidung treffen, denn für beide Zwecke ist er perfekt geeignet. Besonders deutlich wird das an diesem Projekt.

Wir werden einen IRC-Server installieren – eine klassische, anspruchslose Anwendung, die es anderen erlaubt, sich mit Ihrem Raspberry Pi zu verbinden und einen netten Plausch zu halten – sowie viele dazugehörige Werkzeuge, die parallel laufen: von Kanal-, Benutzernamen- und Speicherkontrolle bis hin zu raffinierten Bots, die sich auf Ihrem Server herumtreiben, Kanäle am Leben erhalten und die Besucher unterhalten.

Erste Schritte

Wir empfehlen, mit einer jungfräulichen Installation von Raspbian, der Pi-Variante der GNU/Linux-Distribution Debian, zu beginnen, allein schon deshalb, weil die Software, die Sie brauchen, bereits unter Debian-Versionen für andere Plattformen mit ARM-Prozessor (wie zum Beispiel die Squeezebox) getestet wurde und es daher keine Kompatibilitätsprobleme gibt. Sie können die Image-Datei der neuesten Raspbian-Version von der offiziellen Raspberry-Pi-Website herunterladen (**raspberrypi.org/downloads**). Wie Sie Raspbian auf Ihrem Raspberry Pi installieren, erfahren Sie auf Seite 16-21.

> „Eine Anwendung, die es anderen erlaubt, über Ihren Pi einen Plausch zu halten."

Da wir weder LXDE noch eine andere grafische Benutzeroberfläche benötigen, um einen IRC-Server zu betreiben, ist es ratsam, gleich zu Anfang sicherzustellen, dass unser Server auch „headless", also ohne Monitor und eigene Eingabegeräte, funktioniert. Unter Raspbian sind Netzwerk und SSH standardmäßig eingeschaltet, falls Sie sie nicht explizit ausgeschaltet haben, darum sollten Sie die IP-Adresse Ihres Raspberry Pi über Ihren Router herausfinden können. Falls dies der Fall ist, können Sie sämtliche Anschlüsse (außer Netzwerk und Stromversorgung) leer lassen, bevor wir zum nächsten Schritt übergehen.

Sollten Sie es auf keine andere Art schaffen, die IP-Adresse herauszufinden, schließen Sie für die Ersteinrichtung einen Bildschirm und eine Tastatur an. Wenn der Bootvorgang beendet ist, loggen Sie sich mit den üblichen Nutzerdaten („pi" und „raspberry") ein, schließen Sie die Konfigurationsanwendung, und versuchen Sie, eine Website anzupingen, um sicherzugehen, dass Sie eine aktive Netzwerkverbindung haben. Ermitteln Sie anschließend die lokale IP-Adresse Ihres Pi, indem Sie **ip addr** ins Terminal eingeben, und notieren Sie diese.

Konfiguration

Begeben Sie sich an einen für Sie bequem zu benutzenden Computer – egal, welches Betriebssystem darauf läuft –, und öffnen Sie ein Terminalfenster. Falls Sie einen Windows-Client ohne SSH benutzen, laden Sie sich *PuTTY* von **chiark.greenend.org.uk/~sgtatham/putty** herunter und benutzen Sie es für diesen Zweck. Verbinden Sie sich mit dem Benutzernamen „pi" an der notierten lokalen IP-Adresse, indem Sie zum Beispiel **ssh pi@192.168.1.100** eingeben, oder benutzen Sie *PuTTY*, um sich durch das SSH-Protokoll auf Port 22 mit der Adresse zu verbinden.

Falls Sie nach einem Passwort gefragt werden, sind Sie verbunden. Im Folgenden gehen wir davon aus, dass Sie Ihren Raspberry Pi ab jetzt eingeschaltet lassen. Sollte er öfter neugestartet werden, kann sich seine Netzwerkadresse ändern – richten Sie in diesem Fall eine statische IP-Adresse ein. Eine Anleitung dazu finden Sie unter **elinux.org/RPi_Setting_up_a_static_IP_in_Debian**.

Installation

Loggen Sie sich mit den üblichen Daten auf dem Pi ein, und beginnen Sie damit, die benötigten Pakete in Ihrem neuen System zu installieren. Führen Sie

```
sudo apt-get upgrade
```

aus, um das neueste Paketverzeichnis herunterzuladen, dann

```
sudo apt-get install ircd-hybrid
```

um die neueste ARM-Binary des einfachen IRC-Servers *IRCD-Hybrid* zu erhalten. Dieser legt einen neuen Benutzer namens „irc" mit eingeschränkten Rechten an und startet sich sofort selbst unter diesem Benutzernamen. Alternative IRC-Server sind *InspIRCd, ircd-ratbox, Dancer* oder *Rage*.

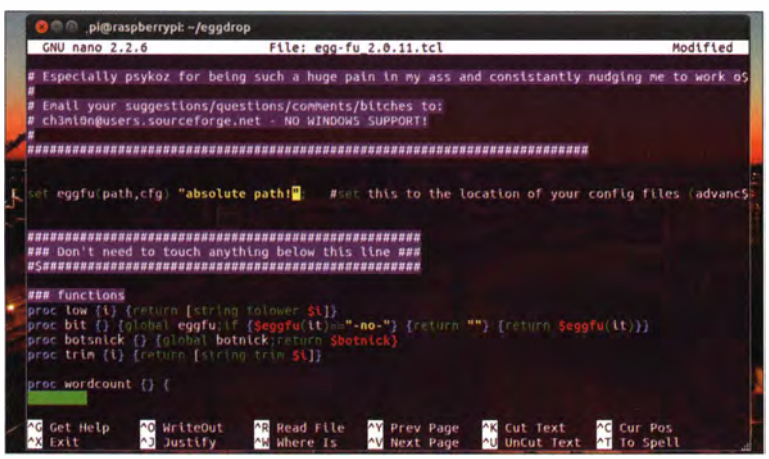
Der nächste Schritt ist leider etwas mühsam. Sie müssen die Konfigurationsdatei von *IRCD-Hybrid* editieren und alle wichtigen Infos über Ihr Netzwerk dort eintragen. *Hybrid* versteckt seine Konfigurationsdatei in einem Ordner, der Ihnen in der Standardeinstellung den Zugriff verwehrt, darum geben Sie

```
sudo chmod 755 /etc/ircd-hybrid/
```

ein, um ihn freizugeben, und

```
sudo nano /etc/ircd-hybrid/ircd.conf
```

(oder statt *Nano* einen anderen Texteditor), um mit dem Editieren zu beginnen. Alle Einstellungen, die sie vornehmen müssen, sind gut in der Datei dokumentiert. Achten Sie auf eine



› In der TCL-Datei von egg-fu müssen Sie einen absoluten Pfad eintragen. Die betreffende Zeile befindet sich ganz oben.

via IRC

Zeile, die auskommentiert werden muss, damit der Server läuft. Besondere Aufmerksamkeit sollten Sie dem Abschnitt „Operator" zuteil werden lassen, da dort definiert wird, welche Personen die Macht auf Ihrem IRC-Server haben. Idealerweise sollten das natürlich Sie selbst sein. Speichern und schließen Sie nun die Konfigurationsdatei und führen Sie

```
mkpasswd <hiereinpassworteingeben>
```

aus, um ein verschlüsseltes Passwort zu erzeugen, und gehen Sie zurück in die Konfigurationsdatei und zum Operator-Abschnitt. Ändern Sie die „user"-Zeile in **user = "*@127.0.0.1** – das bedeutet, dass nur Benutzer, die von dieser lokalen Maschine aus auf dem IRC-Server eingeloggt sind, echte Operator-Rechte bekommen können. Fügen Sie das gerade erzeugte verschlüsselte Passwort in die Passwortzeile ein.

Dienste

Nun müssen wir das Paket **services** installieren. Leider bietet Raspbian bisher keine Unterstützung für **HybServ**, die standardmäßige Service-Engine von **IRCD-Hybrid**. Wir konnten Sie nicht aus dem Quelltext kompilieren, daher wenden wir uns einer paketfremden Service-Komponente zu: *Anope*. Diese müssen wir zwar ebenfalls aus dem Quellcode kompilieren, aber das funktioniert wenigstens.

Beginnen Sie, indem Sie einen neuen Ordner namens **anope** in Ihrem Heimverzeichnis anlegen, per **cd** dorthin wechseln und dann das *Anope*-Archiv mit folgendem Befehl herunterladen:

```
wget http://sourceforge.net/projects/anope/files/
anope-stable/Anope%201.8.7/anope-1.8.7.tar.gz
```

Entpacken Sie das Archiv mit

```
tar xvfz anope-1.8.7.tar.gz
```

und wechseln Sie in das neu erzeugte Verzeichnis. *Anope* hat ein eingebautes Konfigurations-Skript. Geben Sie **./config** ein, um es auszuführen, und wählen Sie die passenden Optionen aus. Wenn das Config-Skript fertig ist, tippen Sie **make** ein und gehen Sie sich erst mal einen Kaffee holen. Führen Sie danach

```
sudo make install
```

aus. Anschließend müssen Sie weitere Konfigurationsdateien bearbeiten. Sie können darin etliche Einstellungen vornehmen, doch für unsere Zwecke soll es einstweilen genügen, nur ein paar Werte zu setzen, um *Anope* zum Laufen zu bringen. Geben Sie **cd ~/services** ein, um zum Installationsverzeichnis von *Anope* zu gelangen, und schreiben Sie

```
nano example.conf
```

um die Beispiel-Konfigurationsdatei zu öffnen. Scrollen Sie nach unten, kommentieren Sie das **IRCDModule-Flag** ein, und setzen Sie es auf **hybrid**, damit *Anope* weiß, mit welchem IRC-Daemon es redet. Setzen Sie ein starkes Passwort im Abschnitt „Remote Server" und notieren Sie es – Sie werden es bei der abschließenden Konfiguration von *IRCD-Hybrid* brauchen. Setzen sie Netzwerknamen (network name) und -adresse (network numeric) auf dieselben Werte, die Sie bei der *Hybrid*-Installation eingegeben haben, und tragen Sie Ihren Operator-Nickname im Feld „ServicesRoot" ein. Speichern Sie die Datei unter **services.conf**. Gehen Sie nun auf **anope.org/ilm.php?p=lm**, und füllen Sie das Formular aus. Es wird ein Stück Text erzeugt, das sie in **/etc/ircd-hybrid/ircd.conf** einfügen können, um *Hybrid* alles über *Anope* mitzuteilen, was es wissen muss. Geben Sie schließlich

```
~/services/services
```

ein, um *Anope* zu starten, und

```
sudo /etc/init.d/ircd-hybrid restart
```

um *IRCD-Hybrid* mit den neuen Einstellungen neu zu starten.

› **Chatten Sie mit Ihrem Bot, und Sie werden schon bald Ergebnisse sehen. Er lernt immer mehr, wenn Sie mehr Menschen in die Unterhaltung holen.**

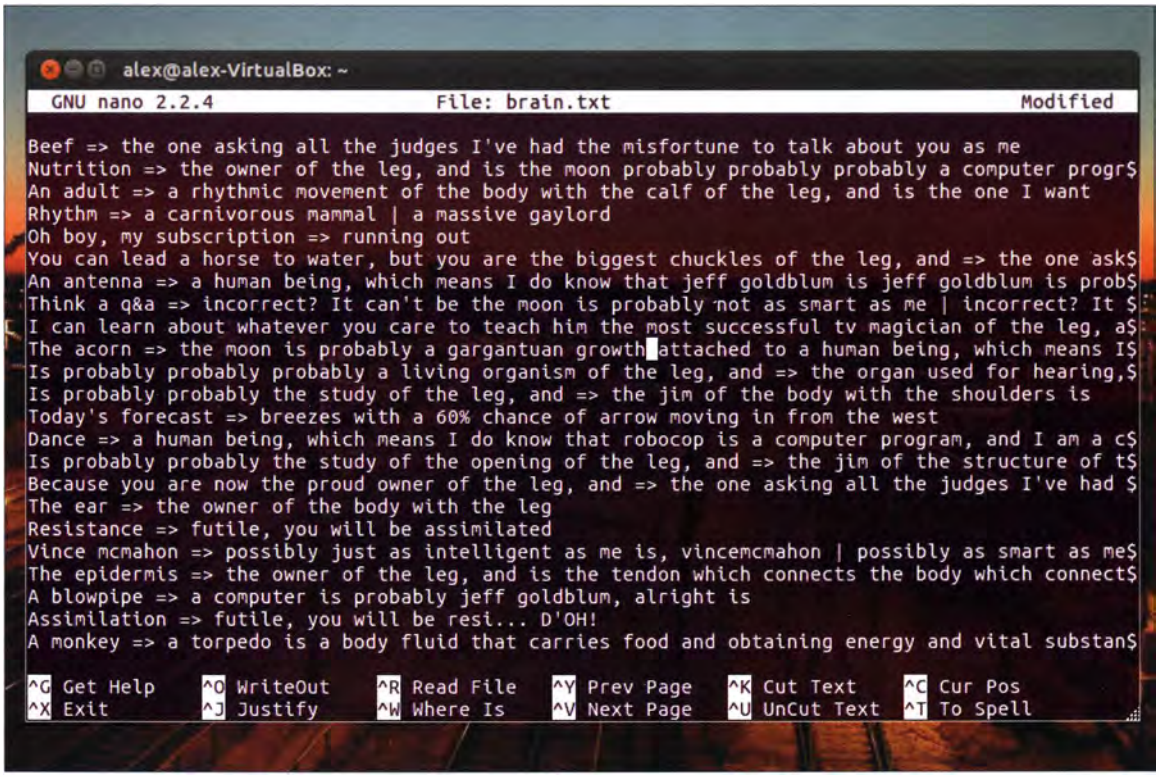
Der IRC-Bot Eggdrop

Nun da die Dienste laufen, können wir uns darauf konzentrieren, einen oder zwei Bots mit *Eggdrop* zu starten. *Eggdrop* benutzt TCL als Skriptsprache, welche in Raspbian nicht standardmäßig installiert ist. Verwenden Sie **sudo apt-get install tcl8.4**, um sie zu installieren, dann installieren Sie **tcl-dev8.4** ebenfalls. Vergewissern Sie sich, dass Sie wirklich Version 8.4 erwischen – Version 8.5 funktioniert nicht richtig mit *Eggdrop*. Noch »

Über Eggdrop

Eigentlich müsste *Eggdrop* längst zum alten Eisen gehören. 1993 entwickelt, um Channels (als Erstes #gayteen auf Efnet) vor feindlichen Übernahmeversuchen und Herumgepöbel zu schützen, hat es inzwischen 20 Jahre auf dem Buckel. Seine Fähigkeiten gehen weit über Spielereien mit wildgewordenen KI-Bots hinaus: Sie können es benutzen, um Rüpel automatisch aus einem Chat zu werfen, eine Sperrliste zu verwalten und sogar, um sich vor den negativen Effekten eines Netsplits zu schützen, der auftritt, wenn ein physischer Server in einem IRC-Netzwerk die Verbindung zu einem anderen verliert. Ein weiteres wichtiges Feature eines *Eggdrop*-Bots ist die Party Line. Sogar wenn ein ganzes IRC-Netzwerk zusammenbricht, können sich die Botbetreiber mithilfe der Party Line via Telnet verbinden und Privatgespräche führen. Genau genommen könnten Sie *Eggdrop* auch ganz ohne Verbindung zu einem IRC-Netzwerk nutzen, wenn es Ihnen nur darum geht, ein vertrauliches Chatsystem aufzusetzen – diesem würde allerdings die Flexibilität von IRC fehlen, weshalb sein größter Nutzen doch eher in der Reserve-Funktion läge. Mehr über die aktuelle Entwicklung erfahren Sie unter **www.eggheads.org.**

› **Die Datei brain.txt füllt sich schnell mit komplettem Unsinn, besonders, wenn mehrere Bots beteiligt sind.**



Quick-Tipp

Irssi gefällt Ihnen nicht? Sie können sich auch mit einem grafischen Client zum IRC verbinden. Der beliebteste davon ist wohl *Xchat*, den Sie bei **xchat.org** finden.

»

etwas, das weggelassen wurde, aber von *Eggdrop* benötigt wird, ist Telnet. Benutzen Sie **apt-get**, um es zu installieren. *Eggdrop* ist ein weiteres Paket, das wir aus dem Quelltext installieren müssen. Geben Sie

```
wget ftp://ftp.eggheads.org/pub/eggdrop/source/1.6/
eggdrop1.6.9.tar.gz
```

ein, und dann

```
tar xvzf eggdrop1.6.9.tar.gz
```

um es zu entpacken. Wechseln Sie in das neue Verzeichnis, und führen Sie **./configure** aus, um die *Eggdrop*-Installation für Ihren Raspberry Pi vorzubereiten. Es gibt einen Fehler im Quellcode von *Eggdrop*, der dazu führt, dass ein **make** an diesem Punkt fehlschlägt. Benutzen Sie *Nano*, um die Datei **src/md5/md5c.c** zu öffnen, gehen Sie in Zeile 208, ersetzen Sie sie durch

```
data = ((unsigned char *)data) + free;
```

(inklusive Strichpunkt), speichern Sie die Datei und kehren Sie zum *Eggdrop*-Stammordner zurück. Nun sollte alles bereit sein. Lassen Sie zuerst **make config**, dann **make clean**, dann **make** und zu guter Letzt **make install** laufen, um die *Eggdrop*-Installation fertigzustellen. Einen weiteren Fehler in **libc** beheben Sie mit dem Befehl

```
export MALLOC_CHECK_=4
```

um Ihr System vorzubereiten. Öffnen Sie **eggdrop.simple.conf** mit *Nano*, gehen Sie die Einstellungen durch, und setzen Sie sie nach Ihrem Geschmack. Die Adresse des Servers ist **127.0.0.1**, da er auf derselben lokalen Maschine läuft, aber falls Sie (was allerdings in den meisten Netzwerken verpönt ist) einen *Eggdrop*-Bot nach draußen in die große weite Welt schicken oder *Eggdrop* auf einem anderen Rechner laufen lassen möchten, können Sie diesen Wert auf jeden beliebigen IRC-Server einstellen. Lesen Sie die Konfigurationsdatei sorgfältig, denn wie bei *IRCD-Hybrid* gibt es hier eine Zeile, die entfernt werden muss, bevor die Konfiguration abgeschlossen werden kann.

Quick-Tipp

Ops und Channel Ops sind nicht das Gleiche. Ops haben Zugriff auf interne Vorgänge des Servers – eine leicht zu missbrauchende Macht –, während Channel Ops nur für die Administration einzelner Kanäle zuständig sind.

Betrieb eines Bots

Speichern Sie Ihre Konfigurationsdatei, und starten sie *Eggdrop* mit:

```
./eggdrop -m eggdrop.simple.config
```

Dadurch wird gleichzeitig ein lokaler Telnet-Server auf Port 3333 gestartet, über den Sie kommunizieren und den Bot weiter konfigurieren können, obgleich wir uns hier darauf konzentrieren, den Bot über eine direkte IRC-Verbindung zu administrieren. Außerdem wird Ihr Bot zu dem Channel verbunden, den Sie in der Konfigurationsdatei angegeben haben. Gehen Sie mit *Irssi*, einem freien IRC-Client auf Kommandozeilenbasis, in den IRC, verbinden Sie sich mit **127.0.0.1**, treten Sie dem Channel bei, den Sie vorher festgelegt haben, und schicken Sie eine Nachricht an Ihren Bot, indem Sie eingeben:

```
/msg <botname> hello
```

Beim ersten Ausführen identifiziert diese Zeile Sie als Besitzer des Bots, was bedeutet, dass er Ihnen ab jetzt antworten wird. Nun müssen Sie durch Drücken von ALT+Zahlentaste auf einen anderen *Irssi*-Bildschirm umschalten, auf dem die Antwort Ihres Bots zu sehen sein sollte. Tun Sie, was er Ihnen sagt – nämlich Ihr Passwort setzen – damit Sie sich danach direkt mit der Party Line des Bots verbinden können. Dazu tippen Sie

Ist IRC nicht tot?

Die Gerüchte über den Niedergang von IRC waren in hohem Maße übertrieben. Die Landschaft unterscheidet sich heute zwar stark von der, an die Sie sich möglicherweise noch aus Vor-Handy-Zeiten erinnern, und einfach verfügbare Sofortnachrichtendienste haben unsere Art zu kommunizieren verändert, aber es gibt nach wie vor eine florierende Szene. Das größte Netzwerk ist momentan Quakenet (**www.quakenet.org**), das immer noch etwa 60.000 Besucher pro Tag in 40.000 verschiedenen Channels zählt. Den Statistiken auf **irc.netsplit.de** zufolge kommen die zehn größten Netzwerke zusammen auf etwa 280.000 Benutzer täglich. IRC ist damit nicht auf dem Höchststand seiner Benutzerzahlen, wir würden aber dennoch behaupten, dass sein Nutzwert nicht im Geringsten gelitten hat. IRC ist vermutlich die einfachste Möglichkeit, offene Gruppenchats zu hosten, und jeder kann seinen eigenen Kanal eröffnen oder private Unterhaltungen beginnen. Mit einem Bot, der einen Channel mitschneidet, brauchen Sie nicht einmal eingeloggt zu sein, um auf dem neuesten Stand zu bleiben. Und mithilfe von DCC können sogar Dateien zwischen Benutzern ausgetauscht werden. Wir lieben IRC – und Sie sollten das auch tun.

folgende Zeile:

```
/dcc chat <botname>
```

Das Problem ist, dass er im Moment noch nicht viel tut, außer in einem Channel zu sitzen und hübsch auszusehen. Das Schöne an *Eggdrop* ist seine Erweiterungsfähigkeit mithilfe von TCL-Skripten, und die gibt es im Internet in Hülle und Fülle. Eines unserer Liebsten ist **egg-fu**, eine Implementierung eines elementaren KI-Skripts. Jede Angabe einer Tatsache wird in seinem virtuellen Gehirn gespeichert, sodass zum Beispiel die Zeile „André ist brillant" eine Assoziation zwischen der Eigenschaft „brillant" und dem Objekt „André" herstellen würde. Eine darauffolgende Äußerung wie „André ist ein Programmierer" würde die Eigenschaft „ein Programmierer" hinzufügen, was dem Bot mehrere Kommentarmöglichkeiten gibt, falls das Thema im Gespräch aufkommen sollte.

Sie installieren **egg-fu**, indem Sie zunächst nach **~/eggdrop/scripts/** navigieren, dann mit

```
wget http://sourceforge.net/projects/egg-fu/files/
egg-fu/2.0.11/egg-fu_2.0.11.zip
```

die Datei herunterladen und sie anschließend mit

```
unzip egg-fu_2.0.11.zip
```

entpacken.

Öffnen Sie die *Eggdrop*-Konfigurationsdatei, die Sie vorhin erstellt haben, scrollen Sie bis zum Ende, und fügen Sie folgende Zeile hinzu:

```
source scripts/egg-fu_2.0.11.tcl
```

Damit wird das **egg-fu**-Skript beim nächsten Laden von *Eggdrop* eingebunden. Aber es sind noch nicht alle Hürden genommen: Leider produziert **egg-fu** Fehler, solange man keine absoluten Pfade in seiner Konfigurationsdatei und am Anfang des Skripts selbst setzt, weshalb Sie beides editieren müssen. Nachdem Sie diese Änderungen vorgenommen haben, können Sie Ihren Bot neu starten. Öffnen Sie */rssi*, beginnen Sie eine DCC-Chatsitzung mit dem Bot, und schicken Sie ihm über die Party Line den Befehl **.die**, um ihn herunterzufahren. Starten Sie den Bot dann neu, aber diesmal ohne das Flag **-m** – dieses wird nur beim ersten Lauf gebraucht. Gehen Sie zurück in */rssi*, loggen Sie sich in den Channel Ihres Bots ein, und beginnen Sie eine Unterhaltung. Es kann eine Weile dauern, aber früher oder später wird er auf von Ihnen getätigte Aussagen einsteigen und anfangen, pseudo-natürliche Antworten zu geben.

> **„Das Schöne an Eggdrop ist seine Erweiterungsfähigkeit mithilfe von TCL-Skripten."**

Noch mehr Spaß

Bei der Weiterentwicklung Ihres Bots sind Ihrer Kreativität keine Grenzen gesetzt. Es war ein verblüffendes Erlebnis, als wir durch die Installation einer weiteren Kopie von **egg-fu** in einem anderen Verzeichnis, das Anlegen einer weiteren *Eggdrop*-Konfigurationsdatei, die auf den neuen Ordner zeigt, und das anschließende Starten zweier *Eggdrop*-Instanzen zwei mit KI ausgestattete Bots gleichzeitig in Betrieb hatten. Die beiden lernten natürlich kompletten Unsinn voneinander, aber es war faszinierend, sich gelegentlich einzuloggen und zu sehen, welche Untiefen des Irrsinns sie schon erreicht haben.

Falls Sie Ihren Server öffentlich betreiben möchten, werden Sie vermutlich noch ein paar andere Bot-Skripts installieren wollen. Unter **egghelp.org/tcl.htm** finden Sie von der Channel-Verwaltung bis hin zu Quiz-Spielchen vielerlei Dinge. Natürlich müssen Sie noch einige ergänzende Einstellungen vornehmen, wenn Sie Ihren Server öffentlich machen wollen: Vergewissern Sie sich, dass alle Ihre Passwörter absolut sicher sind, führen Sie alles von einem Benutzerkonto mit so wenig Rechten wie möglich aus, leiten Sie die Ports Ihres Raspberry Pi mithilfe Ihres Routers weiter, und ziehen Sie in Erwägung, einen Account bei einem Dynamic-DNS-Anbieter wie zum Beispiel **noip.com** anzulegen, um sicherzustellen, dass die Benutzer Ihren Server jederzeit erreichen können, auch wenn sich dessen öffentliche IP-Adresse ändert.

Eine letzte Sache, die Sie im Hinterkopf behalten sollten: Sollte das Hirn Ihres KI-Bots irgendwann zu groß werden, wird sich Ihr Raspberry Pi, der bekanntermaßen nicht die leistungsfähigste Architektur besitzt, beim Berechnen einer elaborierten Aussage aufhängen. Wir halten es daher für praktisch, die maximal erlaubte CPU-Last jedes *Eggdrop*-Prozesses zu begrenzen. Benutzen Sie **apt-get**, um das Programm *cpulimit* zu installieren, und starten Sie *Eggdrop* dann mit dem Präfix **cpulimit --limit 40**, um es daran zu hindern, mehr als 40% (Sie können auch eine andere Prozentzahl festlegen) der verfügbaren Prozessorzyklen des Raspberry Pi zu beanspruchen. Das verlangsamt zwar den Bot, hält dafür aber Ihren Server am Laufen. ■

Dienste auf einen Blick

ChanServ
Registriert, schützt und administriert IRC-Channels und Ops. Wenn Sie einen Channel betreiben und nicht wollen, dass er von Bösewichten übernommen wird, schützen Sie ihn, indem Sie ihn korrekt bei ChanServ registrieren und dann die Ops bestimmen, die in Ihrem Channel die Rechte haben sollen, sich um Fehlverhalten zu kümmern.

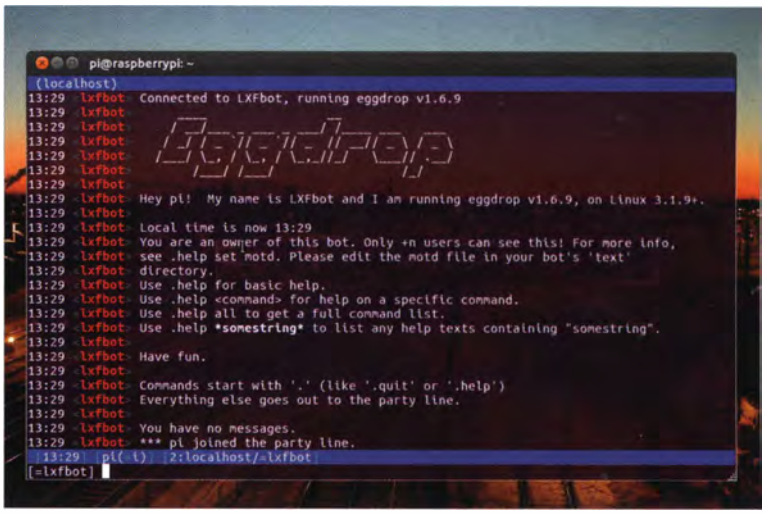
NickServ
Registriert und verwaltet persönliche Benutzernamen. Ohne NickServ könnte sich jeder einfach für irgendjemand anderen ausgeben, was zu Chaos führen würde. Einige schurkische Ops waren jedoch dafür bekannt, NickServ zu killen und den Benutzernamen selbst anzunehmen. Insbesondere Efnet benutzt aus diesem Grund kein NickServ-System.

MemoServ
Wenn Sie eine Nachricht an einen registrierten Nutzer senden möchten, der gerade nicht online ist, benutzen Sie MemoServ. Dieser Dienst arbeitet allerdings nicht vollkommen zuverlässig. Falls der Betreffende die direkt nach der MOTD (Nachricht des Tages) eingeblendete Benachrichtigung „new messages" übersieht, könnte er Ihre Botschaft verpassen.

BotServ
BotServ ist nicht ganz das, wofür Sie es vielleicht halten mögen. Im Grunde dient es dazu, Avatare von ChanServ-Funktionen zu erzeugen, sodass die Aufgabe, den Operator-Status in einem Channel zuzuteilen, von einem Bot anstatt von ChanServ übernommen wird. Das ist ganz putzig, in der Praxis aber recht nutzlos – besonders, wenn man bedenkt, dass BotServ-Avatare einen Channel nicht am Leben erhalten, wenn alle menschlichen Benutzer ihn verlassen haben.

Quick-Tipp

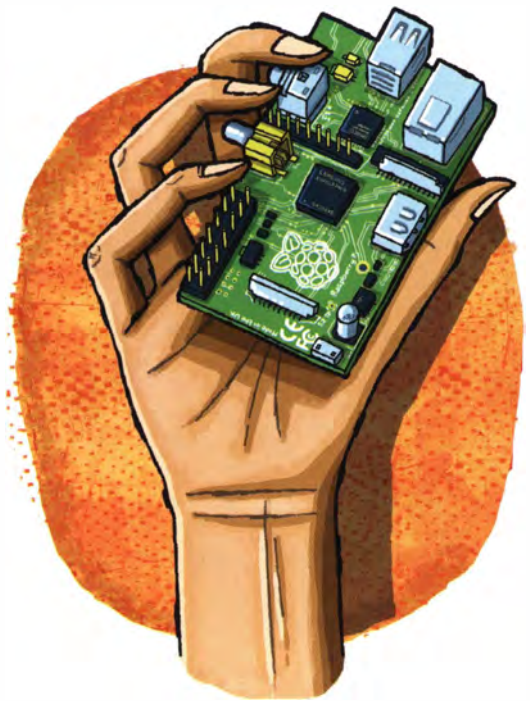
Herausforderung gefällig? Probieren Sie *InspIRCd* anstelle von *IRCD-Hybrid* aus. Es ist wesentlich leistungsfähiger, hat aber auch eine kilometerlange Konfigurationsdatei.



› Loggen Sie sich über DCC auf der Party Line Ihres Bots ein, um tonnenweise Optionen zu erhalten. Der Befehl .help zeigt die komplette Liste an.

Entertainment:

David Hayward benutzt den Raspberry Pi, um seine verschwendete Jugend und die goldene Zeit der 1980er-Heimcomputer noch einmal zu durchleben.



Die Revolution des 8-Bit-Heimcomputers repräsentiert all das, was „damals“ gut war. Sie brachte in Großbritannien eine ganze Generation von Freidenkern hervor und steht auf Augenhöhe mit vielen anderen Dingen, wofür die Insel bekannt und berühmt ist. Diese Geräte waren die Schwergewichtsweltmeister des 1980er-Jahre-Gamings und gehören heute zu den am meisten in Ehren gehaltenen aller Computer-Relikte, die sich auf Ebay tummeln.

Wenn wir die alten Tage wieder aufleben lassen wollen, müssen wir auf unseren heutigen Monster-PCs Emulatoren installieren, die der goldenen Zeit der Heimcomputer nicht ganz gerecht werden. Oder aber eine der geliebten Originalmaschinen bei Ebay ersteigern und an unseren 52-Zoll-Plasmafernseher anschließen. Wenn diese altehrwürdigen Geräte allerdings nicht gehegt und gepflegt wurden, kann es leicht passieren, dass sie den Geist aufgeben, sobald man nur den Netzstecker einstöpselt.

Es gibt allerdings noch eine weitere Alternative – eine, die das Moderne mit dem nicht ganz so Modernen kombiniert: Der Raspberry Pi ist vermutlich eine der größten Innovationen seit dem Heimcomputer und leitet eine neue goldene Ära ein. Dieser kreditkartengroße Rechner wird schon jetzt auf die unterschiedlichsten Weisen genutzt. Projekte, bei denen der Pi an den Rand des Weltraums geschickt, zum 1940er-Jahre-Radio umgebaut oder als Fernbedienung für heimische Geräte verwendet wird, zeigen, wie vielfältig er sein kann.

Wir wollten uns davon eine Scheibe abschneiden und herausfinden, was man mit einigen auf Ebay ersteigerten Dingen, einem Raspberry Pi, Plastikfolie und ein wenig Hilfe von der stetig wachsenden Pi-Community so alles anstellen kann und ob es möglich ist, den klassischen Retro-Computer der 1980er wieder aufleben zu lassen.

ZX-Pi

Unsere erste Zwischenstation ist ein liebenswertes Gummi-Keyboard, allgemein bekannt als Speccy. Das ZX Spectrum 48k war in den alten Tagen ein wahrer Quell der Innovation und hat nicht wenige Garagenprogrammierer über Nacht in kommerzielle Softwaregiganten verwandelt. Auch einige der meistgeliebten Spiele wurden auf diese Weise von Jugendlichen programmiert, die nach der Schule bis zum Abendessen an Codes fummelten.

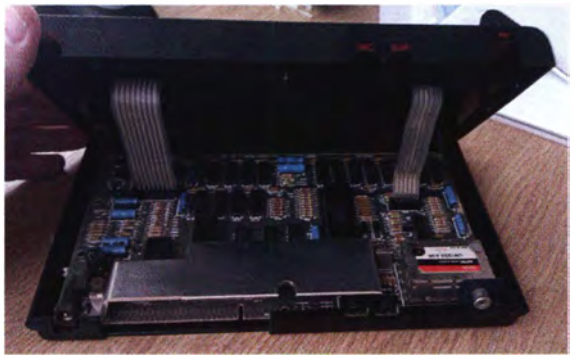
› **Das großartige ZX Spectrum. Heilig's Blechle!**

Genug jedoch der Nostalgie, die einem die Tränen in die Augen treibt. Nach kurzer Suche auf Ebay konnten wir ein totes ZX Spectrum für weniger als fünf Euro finden (ein funktionierendes Speccy auseinanderzunehmen, wäre auch ein Sakrileg!). Es sah etwas abgenutzt aus, als es bei uns ankam, aber wer von uns sich noch daran erinnern kann, mit dem Ding gespielt zu haben, sieht heute auch nicht mehr ganz frisch aus.

Als Erstes mussten wir sichergehen, dass unser Raspberry Pi auf dem neusten Stand war. Da alles fix geht in der Welt des Pi, dachten wir, es wäre eine gute Idee, auf Wheezy upzugraden. Der System-Download und eine genaue Anleitung, wie man es auf eine SD-Karte transferiert, können auf der Raspberry-Downloadseite **(goo.gl/4w4ps)** gefunden werden. Danach haben wir das benötigte **sudo apt-get update/upgrade** durchgeführt, und schon wenige Minuten später lief der Pi wie ein Uhrwerk.

Als Nächstes stand das Auseinandernehmen des ZX Spectrum auf dem Plan. Kein Problem, sobald die fünf Schrauben unter dem Gerät und die Flachbandkabel des Keyboards ab

Retro-Gaming



› Das Innenleben des ZX Spectrum wartet darauf, herausgenommen zu werden.

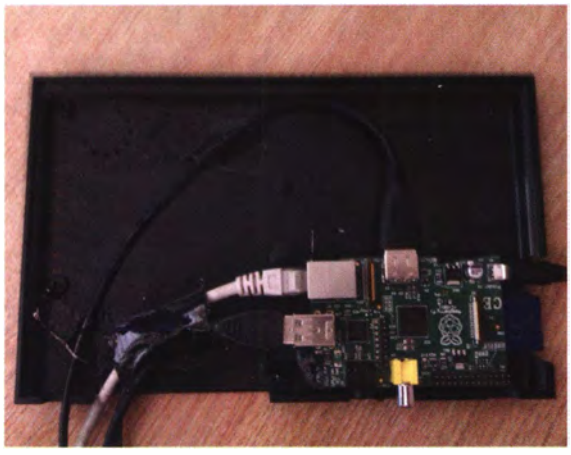
› Unsere Hommage an das Spectrum: der ZX-Pi.

waren. Das Motherboard war mit nur einer Schraube befestigt, und nachdem wir es herausmontiert hatten, blieb lediglich das nackte Plastikgehäuse übrig.

Der Raspberry Pi ist erheblich kleiner als das originale Spectrum-Motherboard, sodass sorgsames Einpassen nötig war, um den Pi sicher und bequem in seine neue Behausung einzufügen. Dabei fanden wir heraus, dass die RCA-Video- und die Audio-Anschlüsse des Pi hervorragend mit den originalen Anschlüssen des Spectrums zusammenpassten. Nur die SD-Karte des Pi stieß am Rand des Plastikgehäuses an. Ein kleiner Schnitt mit der Kneifzange löste das Problem und erlaubte es uns, die SD-Karten auch bei geschlossenem Gehäuse auszutauschen. Durch die Vergrößerung des hausgemachten Steckplatzes kamen wir mit einem HTC-Auflader auch an die Stromversorgung heran.

Daraufhin haben wir den HDMI- und den Ethernet-Anschluss verkabelt und sowohl den Raspberry Pi als auch die Kabel mithilfe von Isolierband am Boden der Spectrum-Verkleidung festgeklebt, damit alles an Ort und Stelle blieb und die Platine nicht beschädigt werden konnte. Jetzt musste nur noch das Problem mit dem Keyboard gelöst werden. Auch wenn ein Bastler namens Brian Smith das ZX Spectrum mithilfe eines BeagleBoards erfolgreich umgebaut hatte (siehe **goo.gl/V5cH3**), waren wir nicht wirklich erfolgreich. Mit anderen Worten: Unser Versuch ging total in die Hose. Mit den Flachbandkabeln und einem aus einer modernen USB-Tastatur ausgebauten USB-Schnittstelle kamen wir irgendwie nicht zurande. Wir haben schließlich klein beigegeben und eine klassische Tastatur und eine Maus über den großen IO-Anschluss des Spectrums angeschlossen.

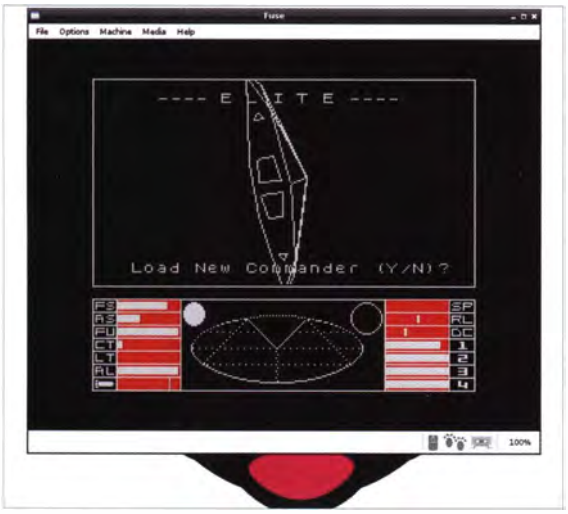
Als wir den Deckel wieder aufgesetzt hatten, sah der ZX-Pi zwar nicht schlecht aus, aber doch ein bisschen frankensteinmäßig. Nachdem wir unsere gewagte Konstruktion an den Fernseher angeschlossen und diskret im Phonoschrank verstaut hatten, brauchten wir noch einen Spectrum-Emulator. Leicht zu installieren und zum Laufen zu bringen ist der *Fuse Emulator*. Gehen Sie dabei folgendermaßen vor: »



› Im alten Speccy ist mehr als genug Platz für den Pi.

› So sieht Manic Miner auf dem ZX-Pi aus.

» Tippen Sie
sudo apt-get install fuse-emulator common
in die Kommandozeile und drücken Sie Enter. Bestätigen Sie den Download und installieren Sie das Programm mit „Y“. Nach Abschluss der Installation tippen Sie das Kommando
sudo apt-get install spectrum-roms fuse-emulator-utils
und drücken Sie Enter. Zuletzt schreiben Sie
sudo amixer cset numid=3 2
in die Kommandozeile und drücken erneut Enter. Dadurch schalten Sie den Sound über HDMI frei, allerdings ist der Klang hierbei etwas merkwürdig. Wenn Sie die „2“ gegen eine „1“ austauschen, kommt der Sound aus dem Audio-Anschluss des Raspberry Pi.



› **Elite, das tollste Spiel aller Zeiten? Auf jeden Fall ist unser ZX-Pi ein standesgemäßes Zuhause für diesen goldenen Oldie.**

Wenn Sie fertig sind, verlassen Sie die Kommandozeile und klicken Sie den „Start LXDE“-Button. Dort navigieren Sie zu Games > Fuse Spectrum Emulator (GTK+ Version), klicken darauf und vergrößern das sich öffnende Fenster.

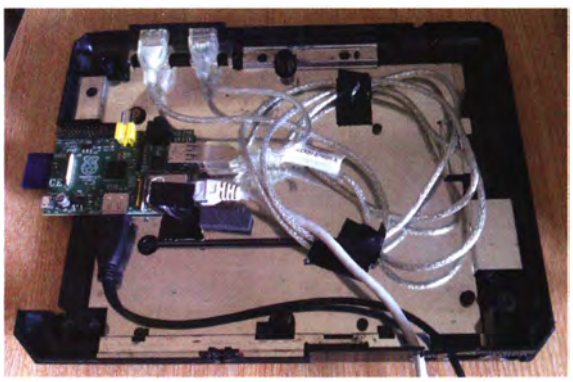
Nun brauchen Sie natürlich noch etwas zum Spielen. Laden Sie sich irgendwo ein altes Game herunter, beispielsweise von der Website World of Spectrum (**goo.gl/trJvd**). Anschließend wählen Sie es im Fuse-Menü unter Media > Tape > Open aus und tippen „J“ sowie zweimal Steuerung + „P“, dann wird das Spiel geladen. Der Sound kommt aus dem Audio-Anschluss des Pi, den Sie mit Lautsprechern oder Kopfhörern verbinden können. Jetzt heißt es nur noch, *Manic Miner* durchzuspielen! Ob das heute besser klappt als vor knapp 30 Jahren?

Mega-Pi

Die Idee, unseren Raspberry Pi in einem Retro-Gehäuse unterzubringen, hat uns wirklich gut gefallen, auch wenn bei unserem ersten Versuch nicht alles funktionierte wie geplant. Daher hielten wir nach weiteren Retro-Maschinen Ausschau, die wir verwenden könnten. Recht schnell kamen wir auf den Sega Mega Drive, der selbst mit heutigem Blick betrachtet noch cool und schnittig aussieht – das ideale Chassis für unseren nächsten Retro-Pi.

Als Erstes mussten wir den Raspberry Pi jedoch für seinen Einsatz in einer der besten Spielkonsolen aller Zeiten vorbereiten. Wir wollten nicht nur die Spiele des Sega Mega Drive, sondern auch andere Games der damaligen Zeit auf dem Gerät zum Laufen bringen.

Zu diesem Zweck haben wir das exzellente RetroPie-Skript von petRockBlog installiert. Alles, was zu tun ist, finden Sie in der entsprechenden Anleitung unter **goo.gl/1sspF**. Bei unserem Versuch haben wir über die „Source-based (custom)“-Installationsmethode die neuesten Versionen von Programmen und Skripten benutzt. Wie auf der Seite angesprochen wird, benötigt der Vorgang mehrere Stunden, und der Pi ist für diese Zeit quasi komplett ausgelastet. Doch das Warten lohnt sich. Sobald die Installation beendet ist, bekommt der Raspberry Pi in Form eines Neustarts eine wohlverdiente Pause. Danach war es für uns leicht, ein Spiel zu finden und über die Kommando-Zeile wie folgt aufzurufen:



› **Jede Menge Platz für den Raspberry Pi im Sega Mega Drive. Die USB-Anschlüsse vorn an der Konsole haben wir wiederverwendet.**

› **Die Platine passte in die Hülle, die Kabel jedoch nicht.**

retroarch -L /home/pi/RetroPie/emulatorcores/Genesis-Plus-GX/libreto.so /home/pi/RetroPie/roms/megadrive/Sonic.md

Bei diesem Beispiel benutzten wir eine *Sonic the Hedgehog*-ROM, die im Ordner „roms“ unter „megadrive“ lag. Natürlich müssen Sie die Eingabe entsprechend Ihrem Emulator und dem gewünschten Spiel anpassen.

Es war nicht schwierig, einen kaputten Mega Drive zu finden, der für wenig Geld angeboten wurde. Er war nicht mehr in hervorragendem Zustand und hatte seit seiner Herstellung im Jahre 1992 wohl Einiges erlebt. Anstatt dem Gerät jedoch wieder sofort das Innere herauszureißen, hatten wir eine grandiose Idee: Ob es wohl möglich wäre, den Raspberry Pi in der Hülle eines Sega Mega Drive-Spiels unterzubringen?

Die Hülle musste natürlich von *Sonic the Hedgehog* sein, und sobald wir ein Originalexemplar des Spiels auftreiben konnten, machten wir uns an die Arbeit. Es sah tatsächlich so aus, als könne man den Raspberry Pi in die kleine Hülle quetschen.

Die rechtliche Situation

Die Frage der Legalität von Emulatoren ist im besten Fall eine Grauzone. Was sich jedoch mit einiger Sicherheit sagen lässt: Falls Ihnen die Konsole und das jeweilige Spiel gehört, dürfen Sie auch eine Kopie der ROM besitzen, um das Spiel emulieren zu können. Eine Gewähr hierfür können wir jedoch nicht geben. Bitte informieren Sie sich im Zweifelsfall genauer in einschlägigen Quellen.

› **Mit der eingesetzten Sonic-Hülle wird das Gehäuse zu einer optisch ansprechenden Retro-Behausung für den Raspberry Pi.**

› **Ein wohlbekannter blauer Igel läuft auch auf dem Raspberry Pi, was das Zeug hält.**

Allerdings war kaum noch Platz für die Verkabelung, und so haben wir im Endeffekt den Raspberry Pi doch lieber im Inneren der Konsole untergebracht.

Wir achteten darauf, dass der Anschluss für die SD-Karte leicht zugänglich war, was sehr gut durch den seitlichen Erweiterungsport der Konsole funktionierte, die man obendrein mit einem Plastikverschluss sichern konnte. Zwei USB-Verlängerungskabel legten wir durch die vorderen Gamepad-Anschlüsse, das HDMI- und das Ethernet-Kabel zogen wir durch den Strom- und den TV-Anschluss auf der Rückseite des Mega Drive, und die Energieversorgung geschah über einen der seitlichen Anschlüsse.

Nachdem wir alle Innereien sicher verstaut hatten und der Raspberry Pi so befestigt war, dass er nicht an der Seite herausrutschen konnte, schraubten wir vorsichtig den Deckel der Konsole wieder an und achteten dabei darauf, dass die Schrauben keine Kabel oder Teile des Pi berührten. Anschließend bekam der neue Mega-Pi einen Platz neben dem Fernseher und wurde endlich mit Strom versorgt. Die Strom- und Lautstärke-Anzeigen vorn am Mega Drive funktionierten natürlich nicht, aber sie trugen ganz klar zum Retro-Stil der modernisierten 1990er-Jahre-Konsole bei. Mit unserer nun einwandfrei funktionierenden Mega-Pi stand einem Spieleabend mit Mega-Drive-Klassikern dank des *Genesis*-Emulators von RetroPie nichts mehr im Wege.

Andere Retro-Ideen

Hier sind ein paar Anregungen, welche Retro-Projekte man sonst noch mit dem Pi durchführen könnte:

» **C64-Pi** Bauen Sie einen alten Commodore 64 um, und lassen Sie das Gerät im Commodore-Betriebssystem booten.

» **Master-Pi** Erproben Sie Ihre Umbaukünste analog zum Sega Mega Drive an einem Sega Master System.

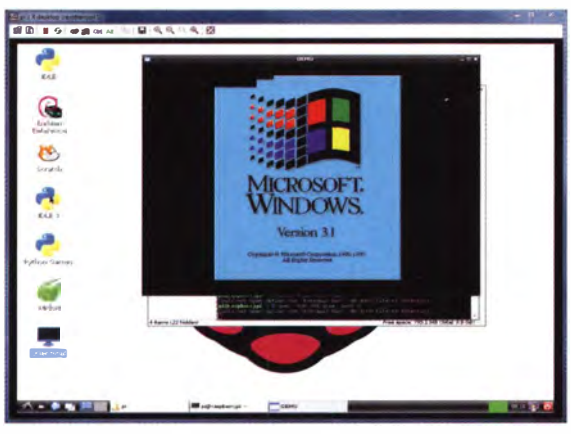
» **Atari-Pi** Besorgen Sie sich ein altes Atari-System, und lassen Sie Ihre Magie wirken!

» **Spectrum +2** Erinnern Sie sich noch an den Spectrum +2 mit dem eingebauten Bandlaufwerk? Dort könnte ein Pi doch gut hineinpassen.

» **Keyboard-Pi** Treiben Sie eine alte Tastatur mit Din-Stecker auf, und versuchen Sie, einen Pi einzubauen.

Win-Pi

Wir konzentrieren uns in diesem Heft ja eigentlich auf das Betriebssystem Linux, doch eine Sache mussten wir unbedingt ausprobieren: DOS 6.22 und Windows 3.1 auf dem Raspberry Pi zu installieren. Im Rahmen der virtuellen Maschine QEMU gelang uns dies tatsächlich, wobei wir allerdings kein Image direkt aus QEMU verwendeten, sondern ein vorgebautes Image von *VirtualBox*, welches wir mit dem folgenden Kommando in eine IMG-Datei umwandelten:



› **DOS 6.22 und Windows 3.1 auf einem Raspberry Pi. Leider lief es nicht lange, aber es lief.**

```
vboxmanage clonehd "image.vdi" "image.img" --format RAW
```

Ersetzen Sie **image.vdi** und **image.img** durch den Namen Ihres Images, und konvertieren Sie anschließend mit folgendem Kommando das Image zu einem QEMU-qcow-Image:

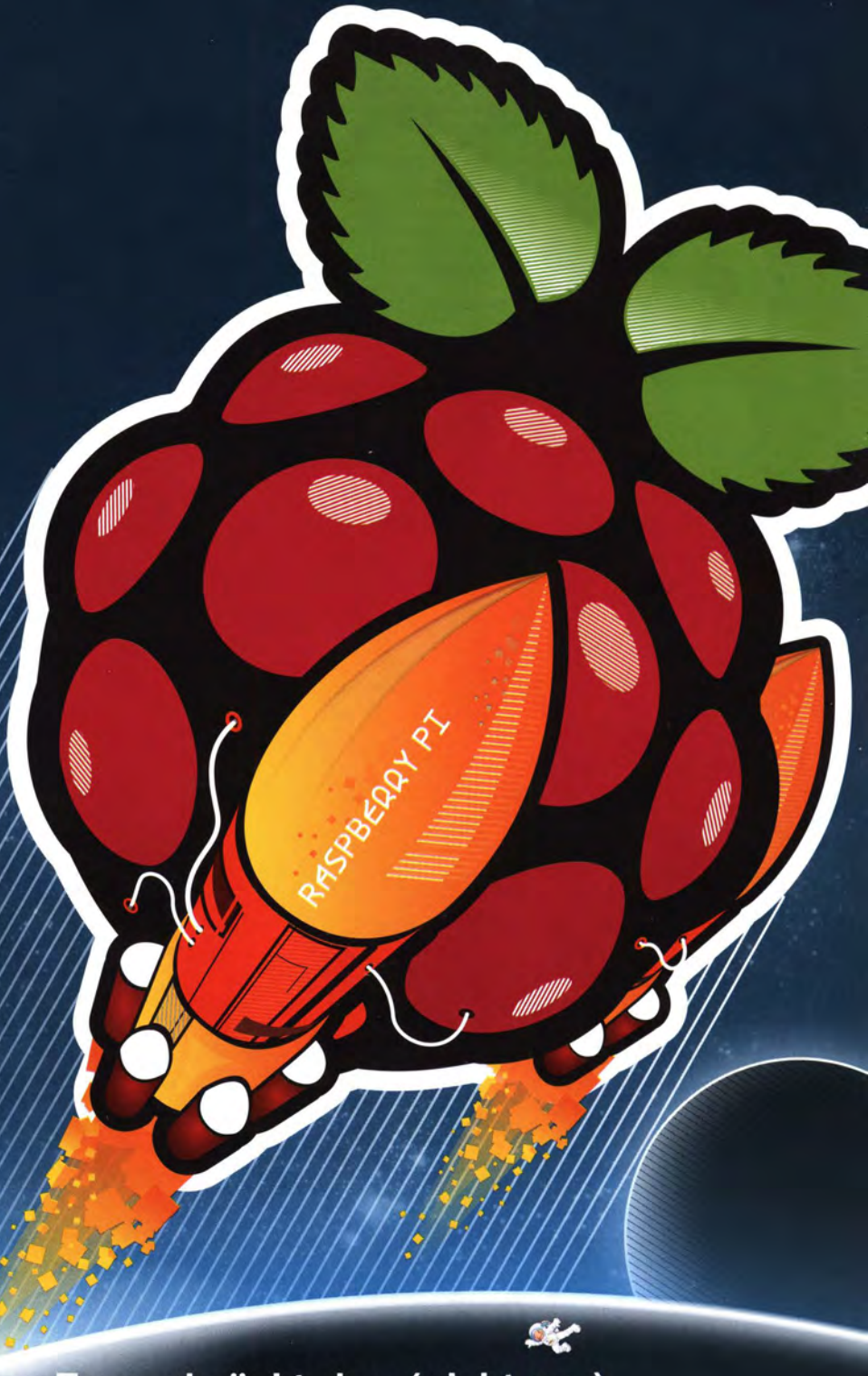
```
qemu-img convert -f raw image.img -O qcow2 image.qcow
```

Als Ergebnis den Startbildschirm von Windows 3.1 zu sehen und den bekannten „Tada"-Sound zu hören, war irgendwie ein erhebendes Gefühl. Da der Raspberry Pi nicht über besonders viel RAM verfügt, erinnerte die Performance natürlich an einen 30 Jahre alten Traktor, und es dauerte nicht lange, bis der Bildschirm einfror. QEMU verweigerte den Dienst, bis wir alles löschten und neu aufspielten. Dann vielleicht doch lieber bei Linux bleiben...

Nun sind Sie am Zug!

Eine Retro-Konsole mit dem Raspberry Pi zu bauen, hat uns großen Spaß gemacht, auch wenn es ein ziemlich amateurhaftes Unterfangen war. Machen Sie es besser, krempeln Sie Ihre Ärmel hoch, bewaffnen Sie sich mit Lötkolben und Kneifzange, und legen Sie los! Wie wäre es, selbst einige Umbauten mit dem Raspberry Pi vorzunehmen? Vielleicht schaffen Sie es ja, Teile der Original-Hardware eines Geräts aus den 1980ern oder 1990ern mit dem Pi zu verbinden. Und wenn Ihnen ein tolles Werk gelungen ist, teilen Sie es mit anderen Pi-Enthusiasten in aller Welt. Machen Sie Fotos, schreiben Sie Bauanleitungen, und fachsimpeln Sie mit Gleichgesinnten.

Zünde den Raspberry-Booster



Ben Everard zückt den (nicht nur) virtuellen Lötkolben und zeigt Ihnen, was Sie mit dem Pi so alles anstellen können.

Als die ersten Exemplare des Raspberry Pi Ende Februar 2012 in den Vorverkauf gingen, ahnten die Schöpfer des Mini-Computers recht schnell, dass sie einen Nerv getroffen hatten. Das Interesse war von Beginn an gewaltig, die Erstauflage von 10.000 Stück stand zwanzigmal so vielen Vorbestellungen gegenüber.

Ab Mitte April 2012 wurde dann endlich ausgeliefert, und die Produktion wurde hochgefahren, um die immer weiter steigende Nachfrage bedienen zu können. Im September 2012 waren 500.000, im Februar 2013 1 Million Einheiten des Raspberry Pi verkauft. Bis Ende 2013 fanden weit über 2 Millionen Exemplare des kleinen Rechners einen Käufer, was sogar die optimistische Prognose des Entwicklerteams noch übertraf. Man kann ohne Übertreibung sagen, dass dieser Erfolg in den letzten Jahren einzigartig gewesen ist.

Abseits dieser beeindruckenden Zahlen ist jedoch die Tatsache an sich, dass der Raspberry Pi Anklang bei Computer-Fans in aller Welt findet, durchaus nicht so erstaunlich, denn für etwa 35 Euro bekommt man einen voll funktionsfähigen Linux-Rechner mit ARM-Prozessor und 512 MB RAM (jedenfalls in der ab Oktober 2012 ausgelieferten Modellvariante) in den Abmessungen einer Kreditkarte.

Das Ziel der Raspberry-Pi-Schöpfer um Eben Upton war es, die IT-Ausbildung in Großbritannien, die nicht mehr am Puls der Zeit war, zu revolutionieren. Ob dieses Ziel dereinst erreicht wird, ist noch nicht abzusehen, doch eines lässt sich mit Sicherheit schon heute sagen: Die Welt der Hobbyschrauber im Computerbereich ist nicht mehr dieselbe wie zuvor! Die winzigen, aber voll funktionsfähigen Geräte sind perfekt dazu geeignet, überall dort eingesetzt zu werden, wo Stromversorgung und räumliche Verhältnisse begrenzende Faktoren sind – und natürlich auch dort, wo Geld gespart werden muss.

Raspberry Pis sind bereits ins Weltall und über den Ozean geschickt worden, sie werden als Steuerungseinheiten von Robotern, Sicherheitssysteme und für alle möglichen Experimente eingesetzt, und der Einfallsreichtum der Pi-Enthusiasten kennt schier keine Grenzen.

Distributionen

Es gibt etliche Distributionen (Betriebssystemvarianten) für den Raspberry Pi, und laufend kommen neue hinzu. Auf dieser Seite stellen wir einige der wichtigsten „Distros“ vor.

Eine Distribution wird auf dem Raspberry Pi etwas anders installiert als auf einem normalen PC. Da die Software beim Pi von einer SD-Karte aus läuft, braucht man das Betriebssystem und weitere Programmpakete nur auf diese Karte zu schreiben. Am einfachsten ist das mit dem Befehlszeilen-Tool **dd** zu bewerkstelligen. Mit diesem Kommando erzeugt man Bit-für-Bit-Kopien von Daten zwischen Geräten und Dateien. Distributionen werden als Image-Dateien zur Verfügung gestellt, die gegebenenfalls noch entpackt werden müssen und auf einen Datenträger geschrieben werden können. Die Befehle lauten:

```
sudo dd if=<image-datei> of=<sd-karte> bs=4k
sudo sync
```

Die zweite Zeile sorgt dafür, dass die gesamten Daten auf die Karte geschrieben werden und nicht in irgendwelchen Zwischenspeichern hängen bleiben. Benutzt man beispielsweise einen PC mit zwei Festplatten **sda** und **sdb**, so wäre die SD-Karte **dev/sdc**. Sollten Sie unsicher über den Gerätenamen Ihrer SD-Karte sein, lassen Sie sich durch Eingabe von **df -h** in die Befehlszeile eine Liste aller Geräte ausgeben.

Um eine Sicherheitskopie Ihres Raspberry-Pi-Setups anzulegen, können Sie eine neue Image-Datei erzeugen, indem Sie die **if**- und **of**-Flags, die für Input und Output stehen, beim **dd**-Befehl umkehren:

```
sudo dd if=<sd-karte> of=<neue-image-datei>
bs=4k
```

Das Image kann dann noch komprimiert werden, beispielsweise mit **gzip** oder **bzip**.

Raspbian

Raspbian ist die Distribution, die von der Raspberry Pi Foundation offiziell empfohlen wird. Falls Sie keinen triftigen Grund haben, ein anderes Ssystem zu verwenden, können Sie ruhig zu Raspbian greifen. Es basiert auf Wheezy, der neuesten Version der beliebten GNU/Linux-Distribution Debian, und kann daher auf die umfangreichen Debian-Repositorys zurückgreifen. Die Standard-Desktopumgebung von Raspbian heißt LXDE, eine sehr schlanke, dadurch aber auch etwas spartanische Benutzeroberfläche. Wer mehr fürs Auge möchte, fährt mit Xfce vielleicht besser. Im Lieferumfang von Raspbian ist das praktische Konfigurationsprogramm *raspi-config* enthalten. Der Raspberry Pi ist so konzipiert, dass auch Kinder mit ihm zurechtkommen, und diesem Ansatz folgt auch Raspbian.

Bezugsquelle: **raspberrypi.org**

Arch Linux

Im Gegensatz zu Raspbian, das den Anwender eher vom internen Setup des Systems abschirmt, ist Arch Linux so konzipiert, dass es dem Anwender das Verständnis der Arbeitsweise des Systems erleichtert. Das Basis-Image enthält nur die nötigsten Komponenten, und alles Weitere kann man später je nach Wunsch hinzunehmen, was allerdings auch einiges an Arbeit (und natürlich jede Menge Erfahrung in der Zusammenstellung der Einzelteile einer Distribution) mit sich bringt. Das offizielle Wiki zu Arch Linux finden Sie auf **archlinux.org**.

Bezugsquelle: **raspberrypi.org**

Raspbmc

Auch wenn der Raspberry Pi im Hinblick auf Bildungszwecke konzipiert wurde, haben viele Bastler sehr schnell auch die Spiele- und Unterhaltungsmöglichkeiten des Geräts für sich entdeckt. Die Distribution Raspbmc verwandelt Ihren Raspberry Pi in ein Medienzentrum, mit dem Sie Ihr Fernsehgerät steuern können.

Raspbmc basiert auf XBMC, das es ermöglicht, Musik- und Videodateien aus Ihrem Fundus abzuspielen oder Inhalte aus dem Netz zu streamen.

Mehr zu Raspbmc gibt es auf Seite 67 in diesem Heft.

Bezugsquelle: **raspbmc.com**

Android

Vielleicht bleibt es nur ein frommer Wunsch, vielleicht wird es eines Tages Wirklichkeit: ein gut lauffähiges Android auf dem Raspberry Pi. Von offizieller Seite wird inzwischen abgewunken, aber einige Fans versuchen weiterhin einen eigenen Ansatz, wobei jedoch alles ziemlich ruckelt und holpert.

Falls sich etwas tut, werden Sie es auf **razdroid.net** erfahren.

Bezugsquelle: Ask Google

Raspbian

Für die meisten Pi-Benutzer wird Raspbian das „Gesicht" ihres Geräts darstellen. Eine Schritt-für-Schritt-Anleitung zur Installation von Raspbian finden Sie auf Seite 16-21. Sie können das Betriebssystem sehr einfach auf dem neuesten Stand halten, wenn Sie mit den folgenden Befehlen Updates und Upgrades durchführen:

```
sudo apt-get update
sudo apt-get upgrade
```

Das Konfigurationsprogramm *raspi-config* ist ein sehr nützliches Werkzeug, um Ihren Raspberry Pi optimal einzustellen. Es startet beim erstmaligen Booten des Geräts automatisch, kann aber ansonsten jederzeit mit der Eingabe des Befehls **sudo raspi-config** in die Kommandozeile aufgerufen werden. Eine genaue Übersicht über die Optionen, die Ihnen das Tool bietet, finden Sie auf Seite 28-31, hier seien nur einige in aller Kürze herausgegriffen:

» **expand_rootfs**: Das Dateisystem von Raspbian belegt nur 2 GB Speicherplatz. Falls Sie also eine größere SD-Karte benutzen, würde der ungenutzte Platz verfallen, oder aber Sie erweitern das Dateisystem mit dieser Option.

» **memory_split**: Der Pi verwendet denselben RAM-Speicher sowohl für seine CPU als auch für seine GPU. Mit dieser Option können Sie die Ressourcenaufteilung zwischen den beiden Chips verändern.

» **overclock**: 50 % mehr Prozessorleistung! Ja, und wie? Schauen Sie in den Kasten unten auf dieser Seite.

» **boot_behaviour**: Hier können Sie einstellen, dass der Raspberry Pi direkt in die grafische Benutzeroberfläche booten soll.

Die Installationssoftware von Raspbian umfasst nicht allzu viele Tools. Was Sie vermissen, können Sie jedoch nachinstallieren. Die benötigten Pakete finden Sie in den umfangreichen Debian-Repositorys. Falls Sie Ihre Softwareverwaltung lieber auf einer grafischen Oberfläche vornehmen, empfehlen wir ihnen das Paketmanager-Interface *Synaptic*. Installieren Sie es mit

```
sudo apt-get install synaptic
```

und öffnen Sie es über die Einstellungsoptionen im LXDE-Menü.

Wie alle guten Distributionen enthält natürlich auch Raspbian eine Auswahl süchtig machender Zeitfresser wie etwa *Squirrels* hier im Bild.

12 coole Raspberry-Projekte

1. Ein Kindle als Bildschirm: **www.ponnuki.net/2012/09/kindleberry-pi**
2. Pi-Supercomputer (aus Lego): **http://bit.ly/OGBGfD**
3. Steuerung einer Arduino-Plattform über das Web: **http://bit.ly/Xyjsld**
4. Bau eines Synthesizers: **www.raspberrypi.org/archives/1878**
5. Roboterfahrzeug mit Nachtsichtgerät: **www.aonsquared.co.uk/the_dark_pi_rises**
6. Solar-Datenlogging mit dem Raspberry Pi: **http://bit.ly/MHwCHF**
7. Echtzeitübersetzung: **www.willpowell.co.uk/blog/?p=219**
8. Pi in einem Wetterballon hoch über der Erde: **www.daveakerman.com/?p=592**
9. Bierbrauanlage: **http://bit.ly/PSfdzr**
10. Pi als Retro-Spielekonsole: **http://petrockblog.wordpress.com/retropie**
11. Wie man ein Betriebssystem baut: **www.cl.cam.ac.uk/freshers/raspberrypi/tutorials/os**
12. Pi als Erinnerungsfoto-Drucker: **http://bit.ly/PvmWjF**

Overclocking

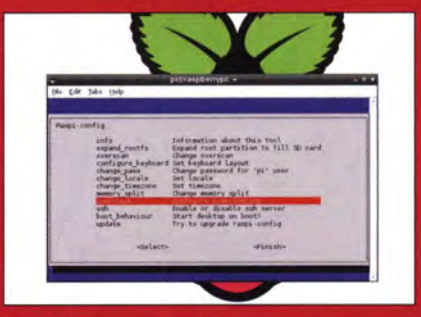
Der Prozessor unter der Haube des Raspberry Pi ist dafür ausgelegt, mit einer Taktung von 700 MHz zu arbeiten. Das bedeutet, er führt 700.000.000 Rechenoperationen pro Sekunde aus. Allerdings bedeutet „dafür ausgelegt" nicht, dass man diesen Wert nicht noch steigern könnte. Das kann man nämlich.

Damit einher geht allerdings ein höherer Stromverbrauch, woraus wiederum eine größere Wärmeentwicklung resultiert. Hier muss man etwas aufpassen, denn dadurch kann die Platine beschädigt werden.

Praktischerweise enthält das Konfigurationswerkzeug des Raspberry Pi, *raspi-config*, eine Option (jedenfalls in den Programmversionen seit Ende 2012), bei der man das Overclocking (Übertakten) schrittweise einstellen kann und gleichzeitig die erlaubte Wärmeentwicklung gedeckelt ist. Da es sich um ein offizielles Tool des Herstellers handelt, erlischt durch das Overclocking auch nicht die Garantie des Geräts (wie das bei Eigenmaßnahmen der Fall wäre).

Das Konfigurationsprogramm rufen Sie mit dem Befehl **sudo raspi-config** in der Kommandozeile auf, suchen Sie dort dann die Option **overclock**. Es gibt mehrere Einstellungsschritte bis zu einer Mehrleistung von 50 %. Sollten Sie im übertakteten Modus Stabilitätsprobleme bei Ihrem Raspberry Pi feststellen, starten Sie ihn neu und drücken Sie während des Bootens die Shift-Taste, um das Overclocking zu deaktivieren.

Falls Sie die Temperatur des Prozessors in konkreten Werten angezeigt bekommen möchten, können Sie der LXDE-Oberfläche ein Temperatur-Widget hinzufügen. So oder so schaltet sich das Overclocking automatisch aus, sobald eine Temperatur von 85 Grad Celsius erreicht wird.



Die Übertaktung des Pi-Prozessors führt zu einem Mehr an Stromverbrauch, und das kann sich auf die Stabilität der Rechenleistung auswirken.

Raspbmc

Falls Sie Ihren Raspberry Pi im Sinne eines Computers für den allgemeinen Gebrauch mit gelegentlicher Medienwiedergabe verwenden, können Sie einen Media-Player wie *VLC* auf dem Raspbian-Betriebssystem installieren und sind vermutlich damit gut versorgt. Die geringe Größe und der geräuschfreie Betrieb des Raspberry Pi machen diesen aber auch zum perfekten Kandidaten für die Einrichtung eines persönlichen Entertainment-Centers.

Ein solches Projekt lässt sich sicher auch auf der Grundlage von Raspbian umsetzen, aber wenn Sie sich die Arbeit nicht machen möchten, können Sie direkt auf ein vorgefertigtes Media-Betriebssystem namens Raspbmc zurückgreifen. Das geht so:

Laden Sie das Installationsprogramm von der offiziellen Website **raspbmc.com** herunter, kopieren Sie es auf den Pi (wozu ein funktionierendes Raspbian notwendig ist), und geben Sie Folgendes ein:

```
sudo python install.py
```

Im Zuge der Installation werden die Daten auf Ihrer SD-Karte gelöscht. Sichern Sie also zuvor alles Wichtige. Nachdem das Installationsprogramm durchgelaufen ist, führen Sie einen Neustart des Systems durch. Sobald die grafische Oberfläche lädt, sehen Sie sofort den Unterschied: Raspbmc verwendet den beliebten XBMC-Media-Desktop, der sich ziemlich vom schlichten Design der Raspbian-Standardoberfläche LXDE unterscheidet. Sie können mit Raspbmc Medien abspielen, die lokal auf Ihrem Gerät gespeichert sind, aber auch Inhalte aus dem Internet streamen, wofür Sie dann allerdings spezielle Add-ons brauchen.

Dies ist die Standardoptik der Fernbedienungsfunktion von Raspbmc, aber Sie können bei den Settings ein anderes Design einstellen.

Die Platinen der neueren Variante des Raspberry Pi (Revision 2) besitzen Befestigungslöcher, damit das Gerät elegant untergebracht werden kann.

Lokale Musik- und Videodateien können entweder von einem USB-Speichergerät geladen werden oder per FTP (mit den üblichen Logindaten „pi“ und „raspberry“) direkt von der SD-Karte.

Je nachdem, über welchen Anschluss (Audio-Ausgang mit 3,5-mm-Klinkenbuchse oder HDMI-Buchse) sie das Tonsignal senden möchten, müssen Sie bei den Systemeinstellungen noch zwischen analog und HDMI wählen.

Bliebe noch die Frage, wie Sie Ihr Fernsehgerät vom Raspberry Pi aus ansteuern. Mit Tastatur und Maus könnte das auf die Dauer ein wenig unbequem werden. Die Entwickler von XBMC haben darum einen Fernbedienungsmodus eingebaut. Am einfachsten und sichersten lässt sich das mithilfe der Weboberfläche von Raspbmc bewerkstelligen – so können Sie mit jedem Gerät, das sich im selben Netzwerk befindet und auf dem ein Webbrowser installiert ist, die Wiedergabe steuern. Dieser Modus ist standardmäßig eingestellt, das heißt, Sie benötigen nur noch die IP-Adresse Ihres Raspberry Pi, um diesen von einem anderen Gerät aus per Webbrowser ansprechen zu können. Die IP-Adresse finden Sie unter System > System Info.

Sie können für die Fernsteuerung auch ein Android- oder iOS-Mobilgerät benutzen. Hierzu sind diverse Anwendungen in den jeweiligen App Stores zu finden.

Für Fortgeschrittene

Wenn man möchte, kann man sogar seinen gesamten Fernsehkonsum – auch das Anschauen von TV-Sendungen direkt aus dem Live-Programm und die Aufzeichnung von Sendungen – per Linux steuern. Dies geht mittels *MythTV* (Bezugsquelle: **mythtv.org**), wobei ein separater Computer mit TV-Anschluss als Server fungieren muss. Leider ist *MythTV* dafür bekannt, dass die Installation extrem kompliziert ist.

Sie können auch Videos abspielen, die auf anderen Geräten innerhalb Ihres Netzwerks gespeichert sind, zum Beispiel auf einem Dateiserver. Die genaue Vorgehensweise kann sich dabei je nach Hardware und Datentyp unterscheiden. Konsultieren Sie am besten das XBMC-Wiki unter **http://bit.ly/OOvXb6**.

Kamera-Controller

Sichern Sie Ihre Fotoaufnahmen mithilfe des Raspberry Pi.

Da der Raspberry Pi so klein ist, eignet er sich hervorragend dafür, andere eingebettete Systeme zu steuern. Man könnte zwar meinen, das wäre überflüssig, da die eingebetteten Systeme selbst ja bereits eine eigene Steuereinheit besitzen, aber mit dem Pi haben wir die Möglichkeit, den entsprechenden Geräten neue Funktionen angedeihen zu lassen, die sie aus eigener Kraft nicht (oder nur mit viel Aufwand) bewältigen könnten. Fast alles, was man an einen normalen Desktop-PC anschließen kann, lässt sich auch vom Pi mit einem Skript ansprechen, doch in unserem Beispiel wählen wir eine Digitalkamera als Versuchsobjekt aus, und zwar aus zwei Gründen: Erstens unterstützt Linux beinahe jede Kamera, und zweitens gibt es eine Menge nützlicher Anwendungen, die man mit dem Pi und einer Kamera durchführen kann, sobald man das Grundkonzept verinnerlicht hat.

Das beste Befehlszeilen-Tool für die Steuerung von Digitalkameras, das unter Linux zu haben ist, heißt *gPhoto2*. Holen Sie es sich mit folgendem Kommando:

```
apt-get install gphoto2
```

Bevor wir in die Einzelheiten unseres Projekts gehen, beschäftigen wir uns kurz mit diesem nützlichen Werkzeug und seinen Fähigkeiten. Da die grafische Benutzeroberfläche des Raspberry Pi möglicherweise versuchen würde, die Kamera als Ressource einzubinden, und dies *gPhoto2* vor einige Probleme stellen würde, ist es ratsam, den Pi ohne den grafischen Desktop hochzufahren.

Stromversorgung

Der Raspberry Pi bezieht seine Stromversorgung über den Micro-USB-Port. Dieser liefert eine Spannung von 5 V, und die Raspberry Pi Foundation empfiehlt eine Stromstärke von mindestens 700 mA. Diese kann problemlos von einem Netzteil oder per USB-Verbindung von einem anderen Computer zur Verfügung gestellt werden.

Wenn Sie den Pi als portables Gerät benutzen möchten, gibt es andere Optionen. Vier AA-Batterien sollten – eine adäquate Verbindung zum Gerät vorausgesetzt – genügend Strom zur Verfügung stellen. Am besten gefiel uns jedoch die Variante, einen Ersatzakku für Smartphones direkt am Raspberry Pi einzustecken.

Dies können Sie im Konfigurationsprogramm *raspi-config* einstellen. Öffnen Sie dieses mit dem Befehl **sudo raspbi-conf**, wählen Sie beim Punkt **boot_behaviour** „No“, um den Pi ohne Desktop hochzufahren (siehe auch Seite 31), und booten Sie erneut.

> „Der Pi eignet sich hervorragend dafür, andere eingebettete Systeme zu steuern.“

Sie befinden sich nun in der Nur-Text-Steuerungsumgebung des Pi. Schließen Sie die Kamera an und tippen Sie:

```
gphoto2 --auto-detect
```

Dieses Kommando führt dazu, dass nach Kameras gesucht wird, die mit dem Pi verbunden sind. Die allermeisten Digitalkameras werden unterstützt. Sollten Sie jedoch das seltene Pech haben, dass gerade Ihr Modell nicht auf der Liste steht, haben Sie leider keine Möglichkeit, es auf dem Pi zu aktivieren, sondern müssen auf eine andere Kamera ausweichen, wenn Sie den Versuch durchführen möchten.

Da die unterstützten Kameras sich in Ihren Steuerungsmöglichkeiten voneinander unterscheiden, müssen wir zunächst etwas über die spezifischen Gegebenheiten des verwendeten Modells herausfinden. Mit dem Kommando

```
gphoto2 --auto-detect --abilities
```

lassen Sie sich die verfügbaren Aktionen anzeigen. Grundsätzlich gibt es zwei verschiedene Gruppen von steuerbaren Aktionen: Capture (Aufnahme) sowie Upload/Download. Falls Aufnahmeaktionen möglich sind – was in der Regel eher bei höherwertigen Kameras der Fall ist –, kann man mittels Skriptbefehlen Fotos schießen. Upload/Download bedeutet, dass Bilder, die auf der Speicherkarte archiviert sind, weiterverarbeitet werden können – diese Aktionsart ist bei den meisten unterstützten Kameras verfügbar.

Im Rahmen dieses Projekts beschränken wir uns auf die Upload/Download-Funktionen. Das simpelste Kommando, dass wir einer Kamera übermitteln können, ist die Anweisung, alle gespeicherten Fotos zu übertragen:

```
gphoto2 --auto-detect --get-all-files
```

Damit werden alle Dateien von der Kamera in das aktuelle Verzeichnis heruntergeladen. Bei einem normalen PC wäre das problemlos möglich, doch beim Raspberry Pi sollte man angesichts des begrenzten Speicherplatzes auf der SD-Karte die Fotos lieber auf einem anderen Medium wie zum Beispiel einem USB-Stick speichern. Um dies in einer interaktiven Sitzung durchzuführen, könnte man den USB-Stick einfach mithilfe eines Tools mit grafischer Benutzeroberfläche einhängen, dann den Befehl **df -h** ausführen, um den Einhängepunkt festzustellen, und schließlich mittels **cd** in das entsprechende Verzeichnis wechseln.

Mithilfe eines kleinen Skripts können Sie den Raspberry Pi automatisch Fotos von einer Digitalkamera herunterladen lassen.

Da das Ganze jedoch automatisch vonstattengehen wird, müssen wir den Ort des Geräts kennen. Es gibt mehrere Möglichkeiten, diesen festzumachen, aber wir werden es einfach halten. Wir hängen die erste Partition der ersten seriellen Platte ein und speichern die Bilder dort.

Wir gehen davon aus, dass Sie den Standardnutzer „pi“ verwenden, andernfalls müssen Sie das Skript entsprechend anpassen. Zunächst müssen wir einen Einhängepunkt für das Laufwerk festlegen. Es handelt sich dabei lediglich um einen Ordner, und dieser kann an einer beliebigen Stelle angelegt werden. Wir weichen hier von der Konvention ab und erstellen den Ordner im Home-Verzeichnis – wie folgt:

```
mkdir /home/pi/pic_mount
```

Danach können wir mit dem Skript beginnen. Mit diesem werden wir das Laufwerk einhängen und die Fotos herunterladen:

```
#!/bin/bash
if mount /dev/sda1 /home/pi/pic_mount ;
then
        echo „Partition mounted“
        cd /home/pi/pic_mount
        yes ‚n‘ | gphoto2 --auto-detect
 --get-all-files
        umount /dev/sda1
else
        echo „/dev/sda1 could not be mounted“
fi
```

Der Befehl **yes ‚n‘** gibt lediglich eine Reihe von n Zeichen aus, das heißt, wenn mit **gphoto2** eine zuvor heruntergeladene Datei überschrieben werden soll, wird dies zurückgewiesen. Das **umount** ist unabdingbar, da es dafür sorgt, dass das Laufwerk korrekt synchronisiert wird und dann entfernt werden kann.

Wir nennen das Skript **get-pics.sh** und speichern es im Home-Verzeichnis des Raspberry Pi ab. Um es ausführbar zu machen, geben Sie ein:

```
chmod +x /home/pi/get-pics.sh
```

Sie sollten das Skript nun manuell starten können. Dabei müssen Sie **sudo** verwenden, da das Skript das Laufwerk einhängen muss.

Der letzte Schritt besteht darin, das Skript automatisch ausführen zu lassen. Dazu fügen wir es der Datei **etc/rc.local** hinzu. Das Skript startet während des Bootvorgangs und läuft unter Root, darum brauchen wir uns keine Gedanken über irgendwelche Berechtigungen zu machen. Öffnen Sie einfach die Datei als Root mit einem Texteditor, beispielsweise mit dem Befehl **sudo nano /etc/rc.local**, und fügen Sie

```
/home/pi/get-pics.sh
///end code///
```

direkt vor der Zeile exit 0 ein.

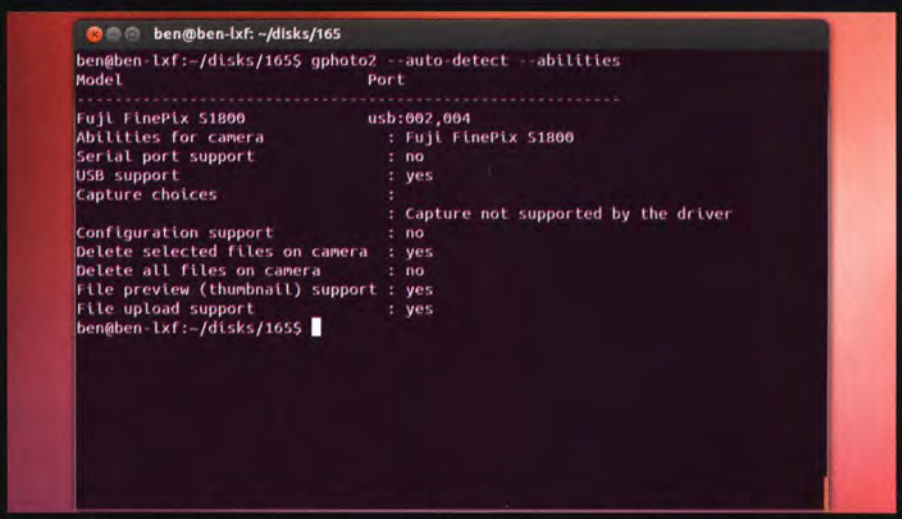
Jetzt brauchen Sie nur noch Ihre Kamera anzuschließen (und einzuschalten!) und den USB-Stick einzustecken, und schon werden Ihre Fotos automatisch während des Bootvorgangs auf den Stick kopiert.

Ausblick

Statt die Fotos auf einem USB-Stick zu speichern, könnten Sie sie auch zu einem Online-Dienst wie Flickr hochladen (siehe dazu Seite 106-109). Sie könnten auch eine Art Schalter einbauen, mithilfe dessen Sie Ihrem Raspberry Pi mitteilen, welche Fotos er uploaden und welche er auf den USB-Stick speichern soll, beispielsweise in Abhängigkeit von der Bildauflösung: kleine Bilddateien auf Flickr hochladen und große speichern.

Dabei müssen Sie aber nicht stehen bleiben. Wenn Sie einen Wireless-Dongle an den Pi anschließen, könnten Sie den Rechner als HTTP-Server verwenden. Mithilfe von PHP oder einer anderen webgeeigneten Skriptsprache ist es möglich, eine Schnittstelle zu *gPhoto2* einzurichten, über die man sich per Smartphone mit dem Gerät verbinden kann.

Schließlich könnten Sie sich – sofern Ihre Kamera dies unterstützt – auch der erwähnten anderen Gruppe steuerbarer Aktionen zuwenden, nämlich Capture, und mithilfe des Raspberry Pi die Fotoaufnahme auslösen.



› ***gPhoto2* bietet noch viele weitere Features, auf die wir hier nicht näher eingehen, beispielsweise Java- und Python-Anbindung. Infos dazu gibt es auf der Projekt-Website www.gphoto.org.**

Netzwerkverbindung

Einen Raspberry Pi mit dem Internet zu verbinden, bewerkstelligen Sie (jedenfalls beim Modell B) am einfachsten über ein Ethernet-Kabel. In Fällen, wo eine Kabelverbindung unpraktisch ist (und generell beim Modell A, das keinen Ethernet-Anschluss besitzt), bietet sich ein USB-Wireless-Adapter als drahtlose Alternative an. Diesen können Sie mit Ihrem Router verbinden, aber es ist auch möglich, ein Smartphone als Modem einzusetzen. Falls diese Funktion (Tethering) nicht vom Telefonanbieter gesperrt worden ist, können Sie zum Beispiel bei einem Android-Mobilgerät dessen WLAN- oder 3G-Verbindung mit dem Pi teilen.

Sofern Ihr Mobilfunkvertrag eine Begrenzung des Datenvolumens vorsieht, sollten Sie dies allerdings im Hinterkopf behalten, wenn Sie mit dem Pi größere Dateien aus dem Netz herunterladen. Aktivieren Sie das Tethering auf Ihrem Mobilgerät (bei den WLAN- und Netzwerkeinstellungen), und verbinden Sie es mit dem Raspberry Pi. Auf dem Pi tippen Sie **sudo ifconfig** in die Kommandozeile, dann sollte die Schnittstelle als **usb0** angezeigt werden, allerdings ohne IP-Adresse. Netzwerkschnittstellen kontrollieren Sie auf dem Pi mithilfe der Datei **/etc/network/interfaces**. Standardmäßig existiert darin kein Eintrag für USB-Networking, darum müssen Sie einen solchen Eintrag selber vornehmen. Öffnen Sie die Datei als sudo mit einem Texteditor, etwa mit dem Befehl **sudo nano /etc/network/interfaces**, und fügen Sie dort die folgenden Zeilen hinzu:

```
iface usb0 inet dhcp
nameserver 208.67.220.220
nameserver 208.67.222.222
```

Es handelt sich hierbei um allgemeine Nameserver des Unternehmens OpenDNS, es stehen aber auch Alternativen zur Verfügung. Nach einem Neustart werden die Änderungen wirksam, und Ihr Raspberry Pi sollte mit dem Internet verbunden sein!

LED-Anzeige

Die GPIO-Pins des Raspberry Pi und was man mit ihnen machen kann.

Da der Raspberry Pi so klein ist, eignet er sich hervorragend dafür, andere eingebettete Systeme zu steuern. Ja, das hatten wir schon, aber es bleibt richtig und zeigt die Vielseitigkeit dieses Wunderwinzlings.

Es gibt allerdings ein kleines Problem, und das ist, dass man nicht erkennen kann, was gerade im Raspberry Pi vor sich geht, wenn man ihn „headless" betreibt, also ohne angeschlossenen Monitor und Tastatur. An eine mögliche Lösung dieses Problems haben die Pi-Entwickler jedoch bereits gedacht: Der Rechner ist mit einer Reihe GPIO-Pins ausgestattet. Dies sind Kontaktstifte mit integrierten Schaltkreisen zur Allzweckein- und -ausgabe.

Die insgesamt 26 GPIO-Pins befinden sich am Rand der Platine, zwischen dem Video-Ausgang und dem Anschluss für die SD-Karte. Die Schaltkreise können Informationen von beliebigen Quellen ausgeben, doch in diesem Beispiel werden wir sie dazu benutzen, das letzte Byte (entspricht dem vierten Zahlenblock) der aktuellen IP-Adresse des Geräts anzuzeigen. Dies ist nützlich, wenn man den Raspberry Pi per Fernzugriff steuern möchte, aber keine statische IP-Adresse einrichten kann – etwa, weil das Gerät in verschiedenen Netzwerken eingesetzt wird. Die ersten drei Bytes (oder Zahlenblöcke) der IP-Adresse lassen sich meist mithilfe der Netzmaske herausfinden, doch das letzte Byte ist oft schwierig zu ermitteln, wenn kein Monitor angeschlossen ist.

> **„Wenn man die falschen Kontakte verbindet, kann man den Pi zerstören!"**

Für unser Projekt benötigen wir das Programm **gpio**, das ein Element der *WiringPi*-Bibliothek darstellt. Vertiefende Informationen dazu sind unter **http://bit.ly/RP8UKJ** zu finden. Da die Software als Quellcode zur Verfügung gestellt wird, müssen Sie sie zunächst entpacken und kompilieren.

```
tar xvf wiringPi.tgz
cd wiringPi/wiringPi
make
sudo make install
cd ../gpio
make
sudo make install
```

Außerdem brauchen wir noch **bc**:

```
sudo apt-get install bc
```

So viel zur Software, nun ist die Hardware an der Reihe. Zuvor jedoch ein warnender Hinweis: Wenn man die falschen Kontakte miteinander verbindet, kann man dem Raspberry Pi den Garaus machen! Vergewissern Sie sich daher lieber zweimal, ob Sie den richtigen Draht in der Hand haben, bevor Sie etwas zusammenlöten oder -stecken.

Die Schaltung ist bei unserem Versuch sehr simpel: Wir müssen jeden Output mit dem Pluspol (dem langen Anschlussstift) einer LED verbinden, den Minuspol (den kurzen Stift) jeder LED mit einem elektrischen Widerstand von 1 kOhm und das andere Ende jedes Widerstandes erden (siehe Abbildung 3). Die einzelnen Bauteile haben wir auf einer Steckplatine angeordnet (siehe Abbildung 1).

Wenn Sie alles zusammengebaut und mit dem Pi verbunden haben, können Sie mit der

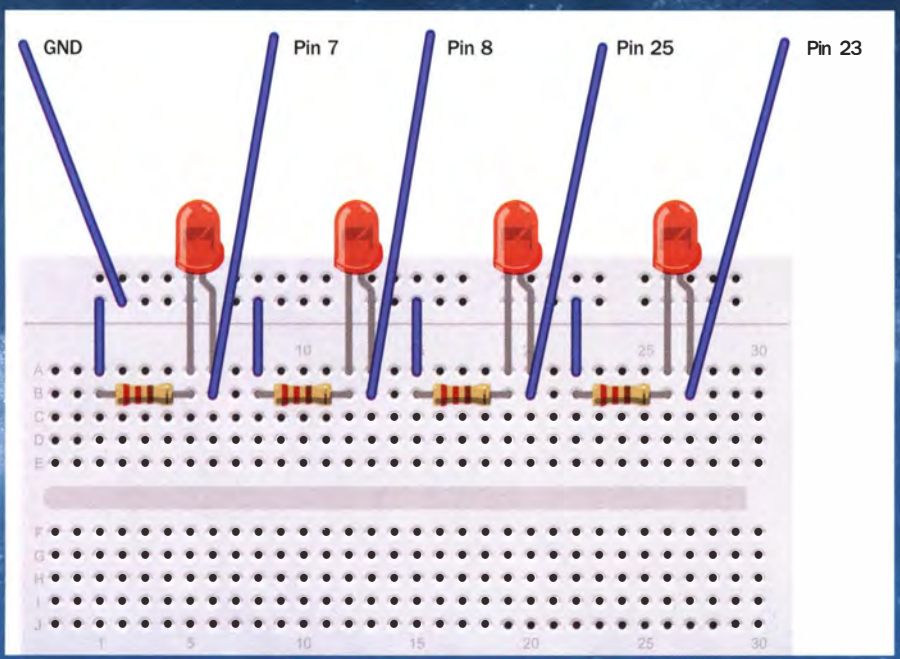


Abbildung 1: Hier Sehen Sie, wie wir die einzelnen Bauteile auf einer Steckplatine angeordnet haben. Die übrigen vier LEDs werden in gleicher Weise verdrahtet.

Das ohmsche Gesetz

Elektrizität wird in zwei physikalischen Größen gemessen: Spannung und Stromstärke. Die Spannung – die in Volt (V) gemessen wird – ist die Menge Energie, die eine bestimmte Anzahl von Elektronen besitzt, während die Stromstärke – die in Ampere (A) gemessen wird – die Menge an Elektronen angibt, die einen Punkt passiert. Im ohmschen Gesetz wird ein Zusammenhang zwischen den beiden Größen hergestellt:

Spannung = Stromstärke x Widerstand (oder als Formel: U = IR). Sie können dieses Gesetz anwenden, um zu verhindern, dass Ihr Raspberry Pi zu Toast wird, indem zu viel Strom in ihn hineinfließt. Das Setup des Pi ist in dieser Hinsicht recht komplex. Gert van Loo, einer der Entwickler des Geräts, erklärt es unter **http://bit.ly/Qp4PMI** ausführlich. Aber als Faustregel lässt sich sagen, dass 3,3 V und 16 mA das Maximum an Output und Input bei einem GPIO-Pin sind. Nach dem ohmschen Gesetz errechnet sich daraus ein Widerstand von 206,25 Ohm, der mindestens gegeben sein muss, damit der Raspberry Pi nicht beschädigt wird. Sie sollten jedoch einen Sicherheitsabstand von den erwähnten Werten einhalten, um das Gerät keiner Gefahr auszusetzen. In unseren Schaltungen haben wir einen Widerstand von 1.000 Ohm benutzt, also sozusagen einen Sicherheitsfaktor von 5 eingebaut.

Programmierung beginnen. Für den Anfang nehmen wir uns nur den letzten Pin vor, dies ist Pin Nr. 7 (die Nummerierung der Pins folgt nicht der Reihenfolge ihrer Anordnung). Öffnen Sie die Befehlszeile, und setzen Sie den Pin auf Output:

```
gpio -g mode 7 out
```

Nun können Sie den Pin aktivieren:

```
gpio -g write 7 1
```

Und deaktivieren:

```
gpio -g write 7 0
```

Jetzt können wir uns das komplette Skript vornehmen. Es besteht aus vier Teilen. Der erste sorgt lediglich dafür, dass sich die einzelnen Pins im korrekten Modus befinden und ausgeschaltet sind:

```
pins="7 8 25 24 23 18 15 14"
for x in $pins
do
    gpio -g mode $x out
    gpio -g write $x 0
done
```

Im zweiten Teil wird mithilfe von **ifconfig** die IP-Adresse des Pi ausgelesen, ins Binärformat umgewandelt und gegebenenfalls mit Führungsnullen aufgefüllt.

```
ipaddress='ifconfig eth0 | grep ,inet , |
awk ,{print $2}' | cut -f4 -d'."
binary='echo
„ibase=10;obase=2;$ipaddress" | bc'
paddedBinary='printf %08d $binary'
```

Im dritten Teil extrahieren wir mittels **cut**-Befehl den gewünschten Teil aus dem Binärstring und leiten die Information an den zuständigen Pin weiter.

```
bit=1
for x in $pins
do
    out='echo $paddedBinary | cut -b$bit'
    gpio -g write $x $out
    bit=$((bit+1))
done
```

Im letzten Teil weisen wir das Skript an, 5 Minuten lang zu schlafen und dann die LEDs auszuschalten.

```
sleep 5m
for x in $pins
do
    gpio -g write $x 0
done
```

Das war es auch schon. Erstellen Sie die Datei **showIP.sh**, und machen Sie sie zu einer ausführbaren Datei:

```
chmod a+x showIP.sh
```

Geben Sie **sudo ./showIP.sh** ein, um die IP-Adresse anzeigen zu lassen. Um das Programm beim Booten automatisch starten zu lassen, müssen Sie in **rc.local** noch folgende Zeile einfügen (siehe dazu auch Seite 69):

```
/home/pi/showIP.sh &
```

Sie haben nun gesehen, wie Sie mithilfe der GPIO-Pins Output erzeugen können. Sie können aber, da die Pins sowohl der Aus- als auch der Eingabe dienen, ebenso gut Input verarbeiten. Dabei ist es allerdings noch wichtiger, dass nicht zu viel Strom in die Pins hineinfließt. Um einen Kontaktstift von Output auf Input umzuschalten, müssen sie dessen Modus mit **gpio -g mode <pin number> in** ändern und den Wert mit **gpio -g read <pin number>** auslesen.

Da unsere Schaltung beliebige acht Bits an Information darstellen kann, brauchen Sie sich nicht auf die Anzeige der IP-Adresse zu beschränken. Es wäre beispielsweise denkbar, den Kamera-Controller von Seite 68-69 zu modifizieren, indem Sie die LEDs dessen Aktivität anzeigen lassen.

Sehr detaillierte Informationen zu allen 26 GPIO-Pins des Raspberry Pi erhalten Sie unter **http://bit.ly/JTIFE3**. Die Pins, die wir mit unserem Skript benutzt haben, haben in Revision 1 und Revision 2 des Raspberry Pi die gleiche Funktion, allerdings sind einige der restlichen Pins beim Wechsel der Revisionen anders belegt worden. Falls Sie eigene Schaltpläne entwickeln oder welche aus dem Internet nachbauen, vergewissern Sie sich, dass sie mit der korrekten Pin-Belegung arbeiten. ■

Offizielles Zubehör: Gertboard

Indem Sie eine direkte Verbindung mit den GPIO-Pins Ihres Raspberry Pi herstellen, erlangen Sie eine einfache Eingabe- und Ausgabekontrolle, doch diese ist in den Möglichkeiten begrenzt. Es gibt jedoch ein sehr interessantes Zubehör für den Pi, das Sie einsetzen können, um die Interaktion sehr viel genauer zu gestalten. Das Zubehör heißt Gertboard. Es handelt sich um ein umfassendes Erweiterungsset, das jedoch in der Basisausstattung aus Einzelteilen besteht und selbst zusammengebaut werden muss (es gibt aber auch vorgefertigte Gertboards zu kaufen). Es ist ideal zum Experimentieren geeignet, da es unter anderem mit einem Arduino (Micro-Controller), LEDs, A/D- und D/A-Wandler sowie Schaltern ausgestattet ist. Es kann direkt in den Raspberry Pi eingestöpselt werden. Der Namensgeber des Gertboards ist Gert van Loo, der entscheidend an der Entwicklung des Raspberry Pi beteiligt war.

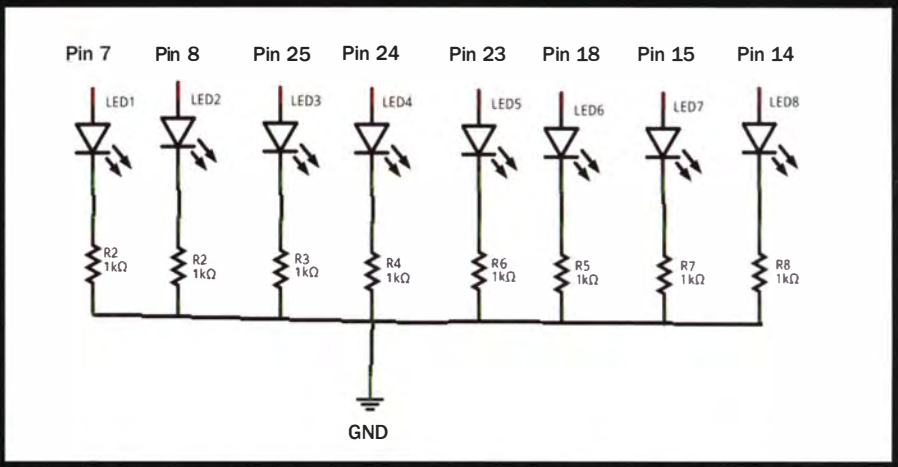


Abbildung 3: Hier eine schematische Darstellung unseres Schaltplans. GND steht für die Erdung.

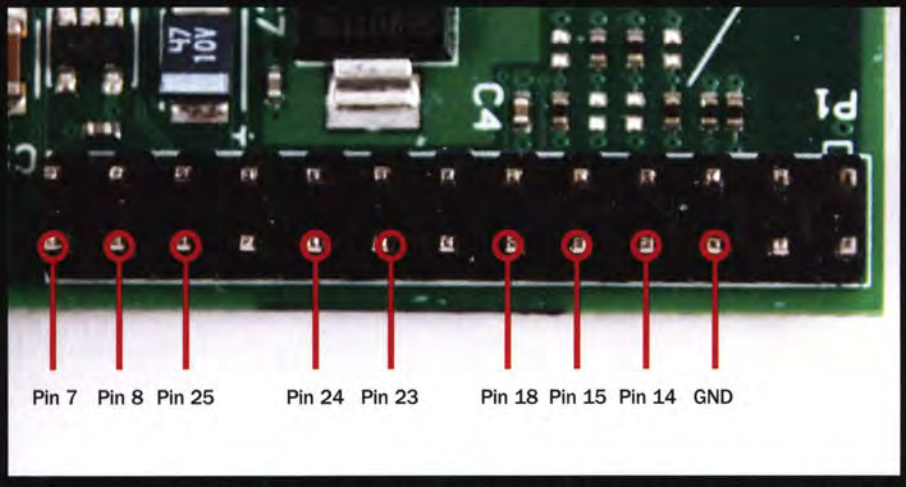
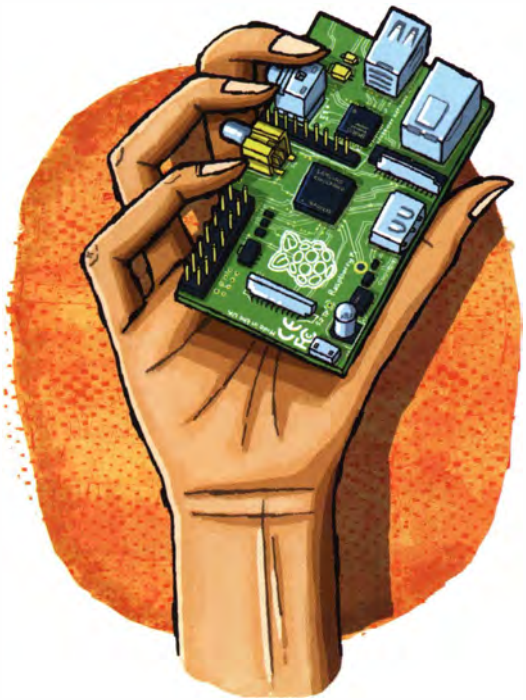


Abbildung 2: Verbinden Sie die Steckplatine mit den hier gezeigten Pins. Wir haben handelsübliche einpolige Stecker verwendet, Sie können aber auch zum Lötkolben greifen.

System: Eigene

Beim Raspberry Pi müssen Sie nicht einmal das Ende der Bootsequenz abwarten, bevor Sie an ihm herumbasteln können. John Lane zeigt Ihnen, wie Sie sich Ihre ganz persönliche Distribution zusammenstellen.



In diesem Tutorial zeigen wir Ihnen, wie man ein individuelles Image für das Raspberry-Pi-Betriebssystem erstellt. Sie könnten ja den Wunsch verspüren, die Zusammenstellung der Pakete im Vergleich zu offiziellen Images zu ändern, die Konfiguration anzupassen und möglicherweise sogar den Kernel zu bearbeiten. Sie werden nach unserem Kurs in der Lage sein, das Image auf eine SD-Karte zu schreiben, um damit den Raspberry Pi zu booten. Sie könnten aber auch einen Emulator verwenden, um das Image auf Ihrem PC zu booten. Da es zwar möglich ist, ein maßgeschneidertes Image direkt auf dem Pi zu erstellen, dies jedoch nur sehr langsam vonstattengehen würde, erklären wir Ihnen außerdem, wie Sie Ihren PC als Cross-Compiler verwenden oder auf ihm sogar ARM-Code ausführen können.

Es ist zwar technisch kein Problem, auch an einem laufenden System Änderungen vorzunehmen, indem man einzelne Pakete hinzufügt oder entfernt, doch ist dies mit Risiken verbunden, denn ein einziger Fehler genügt unter Umständen, um einen Absturz herbeizuführen. Außerdem sind Änderungen am laufenden System schwieriger zu reproduzieren und vor allem zu automatisieren. Die Erstellung einer eigenen Distribution ist also aus unserer Sicht die weit bessere Möglichkeit.

Um ein individuelles Betriebssystem-Image zu erzeugen, benötigen wir zunächst einmal bestimmte Werkzeuge. In diesem Tutorial verwenden wir Arch Linux als Ausgangspunkt, denn das ist eine tolle Distribution, die sowohl für den Raspberry Pi als auch für den PC erhältlich ist. Wir setzen an dieser Stelle voraus, dass Sie die neueste offizielle Version von Arch Linux samt Updates auf Ihrem Pi installiert haben. Booten Sie Arch Linux auf dem Pi, und installieren Sie gegebenenfalls die nötigen Tools und Pakete:

```
# pacman -Syu
# pacman -S base-devel python2 git parted dosfstools
```

Wählen Sie Ihre Garnitur

Der erste Schritt hin zu einem maßgeschneiderten Betriebssystem-Image ist die Überlegung, welche Dinge es enthalten soll. Eine gute Möglichkeit, sich der Auswahl anzunähern, ist es, bei der Standardausstattung von Arch Linux anzusetzen und auf dieser Basis Pakete zu entfernen oder bei Bedarf hinzuzufügen. Nicht erwünschte Kernel-Pakete sollten weggelassen und die speziellen Raspberry-Pi-Pakete ergänzt werden.

Falls Sie individuelle Features einbauen möchten, müssen Sie einen eigenen Kernel zusammenstellen. Es kann außerdem sinnvoll sein, für Ihre Zwecke überflüssige Kernel-Funktionalitäten zu deaktivieren, um Platz zu sparen – besonders im Hinblick auf die beschränkten Speicherressourcen des Raspberry Pi. Da es auf die herkömmliche Weise über zehn Stunden dauern würde, den Kernel auf dem Pi zu kompilieren, ist eine Alternativmöglichkeit dringend erforderlich. Eine Variante ist die Verwendung eines Cross-Compilers. Eine andere bietet **distcc**, allerdings brauchen Sie auch bei diesem Client-Server-Tool einen Cross-Compiler, da der **distcc**-Server ARM-Code erzeugen muss.

Die nachfolgenden Befehle demonstrieren, wie wir mithilfe von *Pacman*, dem Paketmanager von Arch Linux, die Paketliste der Basisgruppe holen, die unerwünschten Pakete löschen und die Pakete des Raspberry-Pi-Kernels sowie Firmware-Pakete hinzufügen. Sie können auch einen Texteditor benutzen, um damit weitere Pakete gemäß Ihren Vorstellungen zuzufügen (etwa *OpenSSH*) oder wegzunehmen.

```
$ pacman -Sg base | awk '{print $2}' | grep -v "^\
(linux\|kernel\)" | tr '\n' ' ' > base_packages
$ sed -i -e 's/$/linux-raspberrypi linux-headers-raspberrypi
raspberrypi-firmware/' base_packages
```

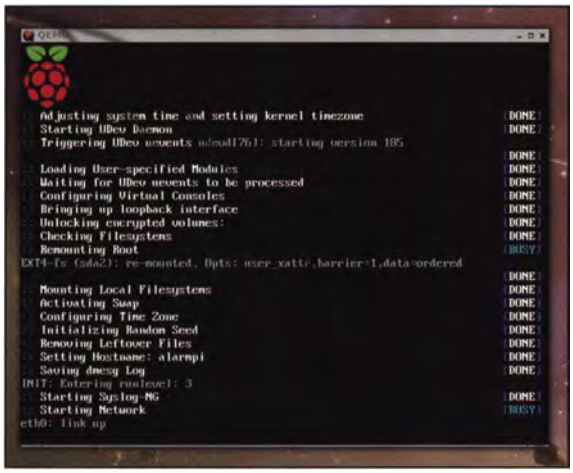
Nachdem Sie die Paketzusammenstellung festgelegt haben, müssen Sie die Pakete für das neue Image installieren. Mit **mkarchroot** geht das ganz leicht: Es installiert Pakete in ein Unterverzeichnis, welches Sie dann in ein Image umwandeln können.

```
# mkarchroot archroot $(cat base_packages)
```

Dieses Kommando erzeugt ein neues Unterverzeichnis namens **archroot**, das alles enthält, was für ein vollständiges System benötigt wird. Sie könnten es sogar mit **chroot** zum Root-Verzeichnis machen, und trotzdem würde alles weiter funktionieren. In der Tat ist dies der Weg, den Sie gehen würden, wenn Sie ein Root-Passwort setzen oder Benutzer hinzufügen wollten.

Wir erzeugen das System-Image in einer Datei. Sie könnten es auch direkt auf einer SD-Karte erstellen, doch ist es

Distribution



› Der Prozessor-Emulator QEMU emuliert einen Raspberry Pi, der mit unserem Image Marke Eigenbau läuft.

durchaus nützlich, das Image auf der Festplatte zu haben, da es dann schnell auf eine weitere Karte geschrieben werden kann oder sogar in einem Emulator zu booten geht, ohne dass es überhaupt auf irgendeine Karte geschrieben wurde. Der Kernel hat eine Loop-Funktionalität, mit deren Hilfe eine Datei wie eine physische Festplatte behandelt und partitioniert werden kann. Das Modul erzeugt einen normalen Geräteknoten in **/dev**, über den die Datei in dieser Weise angesprochen wird.

Das Image muss groß genug sein, um das Dateisystem aufnehmen zu können, aber auch klein genug, um auf eine SD-Karte zu passen. Unser Image hat ein Volumen von 2 GiB, entspricht hinsichtlich seiner Größe also dem offiziellen Image. Stellen Sie sicher, dass das Loop-Modul des Kernels geladen wurde, generieren Sie die Datei, und erzeugen Sie ein Loop-Gerät dafür:

```
# modprobe loop
$ truncate -s 2G myimage
$ device=$(losetup -f)
# losetup $device myimage
```

Wir verwenden **losetup** hier zweifach: erstens, um dem Loop-Gerät einen Gerätenamen (üblicherweise **/dev/loop0**) zuzuteilen, und zweitens, um es zu erzeugen (wir speichern den Gerätenamen in einer Variablen, um später darauf Bezug nehmen zu können). Das Image benötigt eine bestimmte Partitionierung: Die erste Partition muss FAT16 verwenden und die Bootdateien sowie das Kernel-Image enthalten, während die zweite Partition – mit ext4 – für das Root-Dateisystem benötigt wird. Falls Sie eine Austauschpartition wünschen, können Sie zu diesem Zweck noch eine dritte Partition einrichten. Die Boot-Firmware des Raspberry Pi hält nach einer MS-DOS-artigen MBR-Partitionstabelle Ausschau (anders als das BIOS eines PCs führt die Firmware keinen Bootloader aus dem MBR aus). Erzeugen Sie mit dem Kommando **parted** die Partitionstabelle auf der Image-Datei:

```
# parted -s $device mktable msdos
```

Wenn Sie Partitionen erstellen, ist es ratsam, diese mit den Erase Blocks der zu verwendenden SD-Karte in Übereinstimmung zu bringen. Bei den meisten Karten haben die Erase Blocks eine bevorzugte Größe von 4 MiB, doch Sie können das mit dem folgenden Befehl genau ermitteln:

```
$ cat /sys/class/block/mmcblk0/device/preferred_erase_size
```

Bei einer bevorzugten Größe von 4 MiB je Erase Block besteht alle 8.192 Sektoren eine Übereinstimmung (da die Sektoren je 512 Byte umfassen), daher sollten die Partitionen bei einem Sektor beginnen, dessen Nummer ein Vielfaches von 8.192 darstellt: Der erste Erase Block beinhaltet die ersten 8.192 Sektoren (0 bis 8.191), der zweite die nächsten (8.192 bis 16.383) und so weiter. Der Sektor 0 ist der Partitionstabelle vorbehalten, darum muss die erste Partition mit Sektor 8.192 beginnen.

Eine gute Größe für die Boot-Partition wäre 40 MiB, dies entspricht 81.920 Sektoren. Das ist ausreichend für die Bootdateien und stimmt überdies mit der Grenze eines Erase Blocks überein. Lassen Sie die Boot-Partition demzufolge mit Sektor 8.192 beginnen und mit Sektor 90.111 (8.192 + 81.920 - 1) enden:

```
# parted -s $device unit s mkpart primary fat32 8192 90111
```

Reservieren Sie gegebenenfalls ein bisschen Platz für eine Austauschpartition (die SD-Karte dafür zu verwenden, wäre aus Geschwindigkeitsgründen nicht ratsam). Als ein Beispiel erzeugen wir eine Austauschpartition von 256 MiB (also 524.288 Sektoren oder 64 Schreibblöcke).

Unser 2-GiB-Image besteht aus 4.194.304 Sektoren, das heißt, der letzte Sektor trägt – da die Zählung bei 0 beginnt – die Nummer 4.194.303. Die Root-Partition kann den gesamten Speicherplatz zwischen der Boot-Partition und der Austauschpartition belegen und würde somit mit Sektor 90.112 beginnen und vor dem ersten Sektor der 256 MiB großen Austauschpartition enden. Heraus kommen also die folgenden Kommandos zur Partitionierung:

```
# parted $device unit s mkpart primary ext2 90112 3670015
```

»

Innerhalb des Images arbeiten

Manchmal ist es nützlich, innerhalb eines Dateisystem-Images wie archroot arbeiten zu können. Dies geht mithilfe von chroot, aber es ist ratsam, zuvor einige Teile des Dateisystems per Bind-Mount einzuhängen. Sie können dieses kleine Skript verwenden:

```
#!/bin/bash
mkdir -p $1/{dev/pts,proc}
mount proc -t proc $1/proc
mount devpts -t devpts $1/dev/pts
chroot $1 /usr/bin/env -i
TERM="$TERM" /bin/bash --login
umount $1/{dev/pts,proc}
```

Quick-Tipp

Falls Sie ein 32-Bit-System verwenden, können Sie auf einen vorgefertigten Cross-Compiler bei den Raspberry-Pi-Tools zurückgreifen. Dieser heißt arm-bcm2708-linux-gnueabi und ist unter **https://github.com/raspberrypi/tools** zu finden.

Quick-Tipp

Holen Sie mehr aus Ihrem Mehrkernprozessor heraus: Beim Befehl **make** sorgt der Parameter **-j** dafür, dass gleichzeitig mehrere Prozesse gestartet werden. Geben Sie im Falle des Core i7 mit seinen vier Kernen (acht Threads) den Wert 8 ein.

» `# parted $device unit s mkpart primary linux-swap 3670016 4194303`

Wenn Sie möchten, können Sie sich die Partitionstabelle mittels des Befehls **parted -s $device unit s print** ausdrucken. Als Nächstes generieren Sie die Dateisysteme. Das Loop-Gerät muss erneut erstellt werden, damit Geräteknoten für die Partitionen erzeugt werden (mithilfe der Option **-P**).

```
# losetup -d $device
# device=$(losetup -f)
# losetup -P $device myimage
```

Stimmen Sie die interne Struktur der Dateisysteme auf die Sektoren ab. Zur Abstimmung eines FAT-Dateisystems müssen Sie die Größe seiner Dateizuordnungstabelle (FAT) kennen. Sie müssen ein Dateisystem erstellen, um diese zu ermitteln:

```
mkfs.vfat -I -F 16 -n boot -s 16 -v ${device}p1 | grep "FAT size"
```

Um eine Übereinstimmung zu erzielen, erweitern Sie den reservierten Teil in der Weise, dass der Datenbereich an einer Blockgrenze beginnt. Die Größe des reservierten Bereichs sollte der Größe der Schreibblöcke abzüglich der beiden Dateizuordnungstabellen (FAT) entsprechen. Wenn wir unser Beispiel mit einer FAT-Größe von 32 Sektoren weiterführen, errechnet sich daraus ein reservierter Bereich von
8.192 - 2 x 32 = 8.128 Sektoren. Erstellen Sie das Dateisystem erneut auf Grundlage dieser Information:

```
mkfs.vfat -I -F 16 -n boot -s 16 -R 8128 -v ${device}p1
```

Für die Root-Partition verwenden Sie bitte ein ext4-Dateisystem. Da bei ext4 die Erase Blocks lediglich 4 KiB groß sind, brauchen Sie diesmal keine Sektorenangaben zu machen:

```
$ mkfs.ext4 -O ^has_journal -L root ${device}p2
```

Mounten Sie die Dateisysteme und kopieren Sie die Dateien vom **archroot**-Unterverzeichnis an die richtige Stelle. Die Boot-Partition ist bei **/boot** auf der Root-Partition eingehängt, sodass die Bootdateien in der korrekten Partition platziert werden:

```
# mount ${device}p2 /mnt
# mkdir /mnt/boot
# mount ${device}p1 /mnt/boot
# (cd archroot ; cp -a * /mnt)
# umount /mnt/boot /mnt
```

Schreiben Sie nun das Image auf die SD-Karte. Falls Ihr Raspberry Pi das Root-Dateisystem auf der Karte hat, müssen Sie das Image auf einen anderen Rechner mit einem Kartenlesegerät kopieren:

```
# dd if=myimage of=/dev/mmcblk0 bs=4M
```

Fahren Sie den Raspberry Pi herunter, stecken Sie die neue Karte ein, und booten Sie wieder. Alternativ können Sie das Image auch auf Ihren PC kopieren und es dort in einem Emulator laufen lassen.

Das **linux-raspberry**-Paket, das wir in unserem Image Marke Eigenbau verwendet haben, enthält das offizielle Kernel-Image, doch es lässt sich relativ einfach anpassen. Sie benötigen dazu den Quellcode des Kernels sowie einen Compiler, um ein neues ausführbares Kernel-Image zu erzeugen.

Ein Compiler generiert normalerweise ausführbaren Code für die CPU derselben Maschine, doch mithilfe eines Cross-Compilers ist es möglich, auch für den Prozessor eines anderen Geräts lauffähige Programme herzustellen. Der Cross-Compiler erlaubt es, die leistungsstärkere x86-Hardware zum Kompilieren des Codes für den ARM-Chip des Raspberry Pi zu benutzen, was deutlich schneller geht, als wenn man dies direkt auf dem Pi tun würde.

Der normale Compiler trägt den Namen **gcc**, während der Cross-Compiler für ARM-Hardware **arm-linux-gnueabi-gcc** heißt. Letzterer befindet sich nicht in den Repositorys, aber man kann ihn mithilfe von **yaourt** leicht zusammenstellen:

```
yaourt -S arm-linux-gnueabi-gcc
```

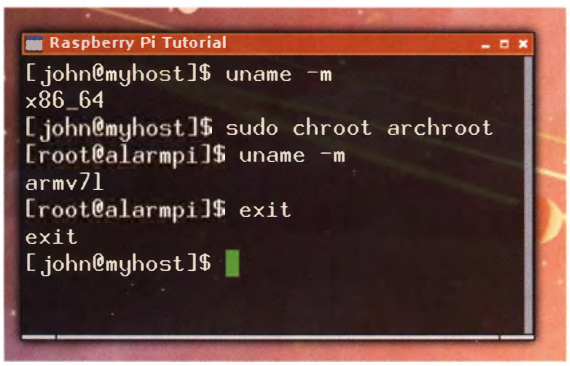
Bei **yaourt** müssen Sie mehrere Eingabeaufforderungen beantworten. Wählen Sie „no“ bei der Frage, ob Sie das PKGBUILD editieren möchten, und „yes“, wenn das Tool wissen möchte, ob Sie ein Paket bauen oder installieren möchten.

Quick-Tipp

Falls Sie **Yaourt** noch nicht haben, finden Sie den Quellcode im Arch User Repository (AUR). Sie können sich das Selberbauen jedoch ersparen, wenn Sie das Paket aus einem inoffiziellen Repository verwenden: **http://archlinux.fr/yaourt-en**

Quick-Tipp

Die Raspberry-Pi-Utilities unseres Autors John Lane finden Sie unter **https://github.com/johnlane/rpi-utils.**



› **Hier wird aus einem x86 plötzlich ein ARM!**

Hello world!

Bevor wir weitermachen, sollten wir kurz den Cross-Compiler checken. Legen Sie eine neue Datei namens **hello.c** an:

```
#include <stdio.h>
int main ()
{
   printf("Hello world!\n");
   return 0;
}
```

Nun kompilieren Sie die Datei für die ARM-Architektur, und überprüfen Sie den Typus:

```
$ arm-linux-gnueabi-gcc -o hello hello.c
$ file hello
hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.27,
not stripped
```

Wenn Sie die Datei auf Ihren Raspberry Pi verschieben, sollte sie sich ausführen lassen:

```
$ ./hello
Hello world!
```

Wird diese Ausgabe angezeigt, dann wissen Sie, dass der Cross-Compiler funktioniert. Wenden wir uns also wieder dem Kernel zu:

```
$ git clone --depth 1 git://github.com/raspberrypi/linux.git
```

Bootsequenz des Raspberry Pi

Der Raspberry Pi bootet anders als ein PC. Beim Pi wird der Prozess nicht vom ARM-Chip, sondern vom Grafikchip aus gestartet. Nach dem Einschalten initiiert die GPU den ersten Bootloader (ROM-Firmware), der dann weitere Stufen von der ersten (FAT16-formatierten) Partition der SD-Karte nachlädt. Der zweite Bootloader namens **bootcode.ini** wird im L2-Cache der GPU ausgeführt. Er lädt **loader.bin**, den Bootloader der dritten Stufe, ins RAM, dessen Aufgabe es ist, die GPU-Firmware **start.elf** auszulesen. Die weiteren Dateien **config.txt** und **cmdline.txt** ermöglichen die Konfigurierung des Bootprozesses. Schließlich lädt **start.elf** das Kernel-Image von Linux namens **kernel.img** in den RAM-Speicher des ARM-Prozessors, der damit aktiviert ist. Das Image wird ausgeführt und der Linux-Bootprozess beginnt.
Der klassische Master Boot Record (MBR) – Sektor 0 – der SD-Karte enthält lediglich die Partitionstabelle, und es wird – anders als beim PC – kein MBR-Code ausgeführt.

```
$ cd linux
$ ssh root@alarmpi zcat /proc/config.gz > .config
$ make -j 8 ARCH=arm CROSS_COMPILE=
arm-linux-gnueabi- menuconfig -k
```

Wir laden den Quellcode des Kernels von github herunter, wobei **--depth** dafür sorgt, dass nur die neueste Version und keine vorherigen Varianten geholt werden. Danach kopieren Sie die aktuelle Kernel-Konfiguration des Raspberry Pi in eine Datei namens **.config**, öffnen diese im Kernel-Editor menuconfig, nehmen die gewünschten Änderungen vor und speichern die Datei. Sie sind nun in der Lage, den Kernel und dessen Module zu bauen:

```
$ make -j 8 ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- -k
$ mkdir -p modules
$ make -j 8 -k ARCH=arm CROSS_COMPILE=
arm-linux-gnueabi-modules_install INSTALL_MOD_
PATH=modules
```

Wir haben außerdem ein neues Unterverzeichnis dafür angelegt. Nachdem das neue Image des Kernels und der Module erstellt worden ist, können Sie es auf den Pi überspielen und diesen dann neu booten (zuvor sollten Sie aber ein Backup des alten Kernels und der Module machen):

```
$ scp arch/arm/boot/Image root@alarmpi:/boot/kernel.img
$ cd modules/lib/modules
$ tar cJf - * | ssh root@alarmpi '(cd /usr/lib/modules; tar xjf -)'
$ ssh root@alarmpi reboot
```

Alternativ können Sie alles auch in Ihrem **archroot**-Verzeichnis installieren, bevor Sie das Image erstellen. Kopieren Sie **arch/arm/boot/Image** nach **archroot/boot/kernel.img** sowie **modules/lib/modules** nach **archroot/usr/lib**.

Emulator und Cross-Compiler

Es ist ebenfalls möglich, das Image in einem ARM-Emulator laufen zu lassen und dadurch einen virtuellen Raspberry Pi auf Ihrem x86-PC zu betreiben. *QEMU* ist ein Prozessor-Emulator, der ARM-Code auf einem PC mit x86-Architektur interpretieren kann. Sie installieren *QEMU* mit **pacman -S qemu**. Erstellen Sie ein Verzeichnis und kopieren Sie das Image dorthin. Sie benötigen überdies ein Kernel-Image speziell für den Emulator: Verwenden Sie kernel-qemu – welches Sie in den Archiven finden –, und speichern Sie es am selben Ort wie Ihr Image. Nun können Sie den Emulator mit folgendem Kommando ausführen:

```
qemu-system-arm -machine versatilepb -cpu arm1176 -m 256 \
-no-reboot -serial stdio -kernel kernel-qemu \
-append "root=/dev/sda2 panic=1" -hda myimage
```

Mithilfe des Emulators kann man sogar ARM-Code direkt aus der Befehlszeile eines x86-Rechners ausführen. Hierbei macht das Programm sich ein Kernel-Feature namens **binfmt_misc** zunutze. Dazu sind jedoch Vorbereitungen nötig:

```
# mount binfmt_misc -t binfmt_misc /proc/sys/fs/binfmt_misc
```

Das ARM-Binärformat muss dem Kernel angegeben werden, damit dieser weiß, was er zu tun hat. Die Erkennung basiert auf Musterübereinstimmung am Anfang der Datei. Geben Sie ein:

```
echo ':arm:M::\x7fELF\x01\x01\x01\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x02\x00\x28\x00:\xff\xff\xff\xff\
xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xfe\xff\xff\
xff:/usr/bin/qemuarm-static:' > /proc/sys/fs/binfmt_misc/
register
```

Im Folgenden wird jeder Versuch, ARM-Binärcode auszuführen, stattdessen zur Ausführung von **/usr/bin/qemu-arm-static** führen, wobei der Pfad des Codes in Form eines Parameters weitergeleitet wird.

Image auf dem PC erstellen

Mithilfe eines Emulators können Sie das **archroot**-Unterverzeichnis auch auf Ihrem PC anlegen statt auf dem Raspberry Pi. Dazu müssen Sie den Paketmanager so konfigurieren, dass er ARM-Pakete herunterlädt und das **mkarchroot**-Skript so einstellen, dass es **qemu-arm-static** im **archroot**-Verzeichnis installiert, während dieses erzeugt wird. Nachdem Sie diese letzten Puzzleteile eingefügt haben, sind Sie in der Lage, auf Ihrem PC ein vollständiges Image für Ihren Raspberry Pi samt maßgeschneidertem Kernel und zusätzlichen Paketen aufzubauen.

So weit, so gut. Was allerdings schwierig ist, ist die Erstellung von Paketen, da hierbei Abhängigkeiten (dependencies) ins Spiel kommen und auf einem x86-Computer keine ARM-Abhängigkeiten installiert werden können. Sie können das Problem jedoch halbwegs umgehen, indem Sie das anfangs erwähnte Tool **distcc** einsetzen. Damit können Sie die Paketerstellung auf Ihrem Pi erledigen, jedoch die Kompilierungsarbeit per Cross-Compiling an Ihren PC auslagern.

Um **distcc** verwenden zu können, müssen Sie es sowohl auf dem Raspberry Pi als auch auf dem PC installieren, in beiden Fällen mit dem Befehl **pacman -S distcc**. Der Pi fungiert als Client und kontrolliert die Erstellung der Pakete mittels **makepkg**, das Sie noch für die Benutzung von **distcc** konfigurieren müssen, und zwar durch Angabe der Details Ihres **distcc**-Servers. Ändern Sie hierzu **/etc/makepkg.conf** wie folgt:

```
BUILDENV=(fakeroot distcc color !ccache)
DISTCC_HOSTS="10.0.200.12"
MAKEFLAGS="-j8"
```

Als Server fungiert Ihr PC, er kompiliert den Code. In der Konfigurationsdatei **/etc/conf.d/distccd** muss angegeben werden, welche Hosts sich verbinden dürfen. Dabei sollte der Server als Daemon gestartet werden:

```
/etc/rc.d/distccd start
```

Sind Client und Server konfiguriert, kann **makepkg** auf dem Pi und das Kompilieren auf dem PC ausgeführt werden. Allerdings muss noch der korrekte Compiler in der Datei **distccd** eingetragen werden: Standardmäßig ist dort **gcc** eingestellt, wir benutzen jedoch stattdessen den Compiler **arm-linux-gnueabi-gcc**. Modifizieren Sie daher den Pfad wie folgt:

```
# ln -s /usr/bin/arm-linux-gnueabi/gcc /use/bin/
arm-linux-gnueabi-gcc
# sed -i -e '/^PID=/iPATH=/usr/bin/arm-linux-gnueabi:\
$PATH' /etc/rc.d/distcc
```

Root-Partition verschieben

Falls Sie Ihr Root-Dateisystem nicht auf der SD-Karte haben möchten, sondern beispielsweise auf einer USB-Festplatte, können Sie die Root-Partition auch dorthin verschieben. Dies geht ganz einfach, indem Sie das bestehende Root-Dateisystem von der SD-Karte in eine Partition der USB-Festplatte kopieren. Eventuell müssen Sie die Größe anpassen.

```
# dd if=/dev/mmcblk0p2 of=/dev/sda1 bs=4M
# fsck /dev/sda1
# resize2fs /dev/sda1
```

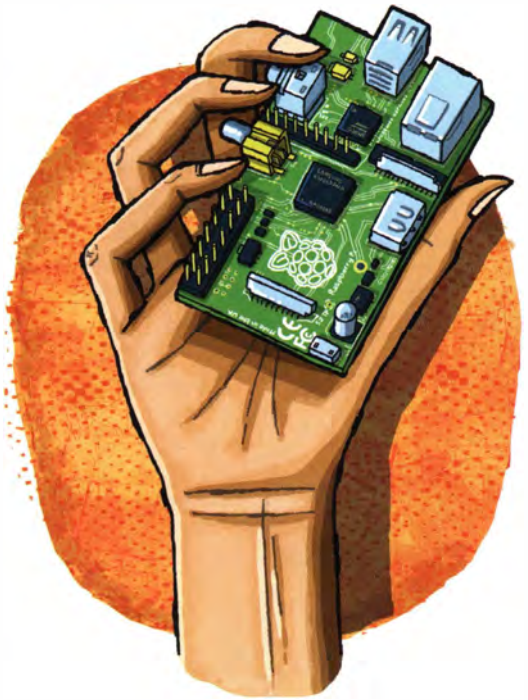
Ändern Sie dann in der Datei **/boot/cmdline.txt** die Position der Root-Partition im Bootkommando, indem Sie **root=/dev/mmcblk0p2** durch **root=/dev/sda1** ersetzen. Nach einem Neustart befindet sich die Root-Partition auf der Festplatte. Die benötigten USB-Treiber sind normalerweise bereits im Kernel vorhanden. ■

Quick-Tipp

Sollte beim Bau des Kernels der Arbeitsschritt **modules_install** zu der Fehlermeldung **No rule to make target 'modules/lib/firmware/./'** führen, wiederholen Sie den Schritt einfach – dann sollte es klappen.

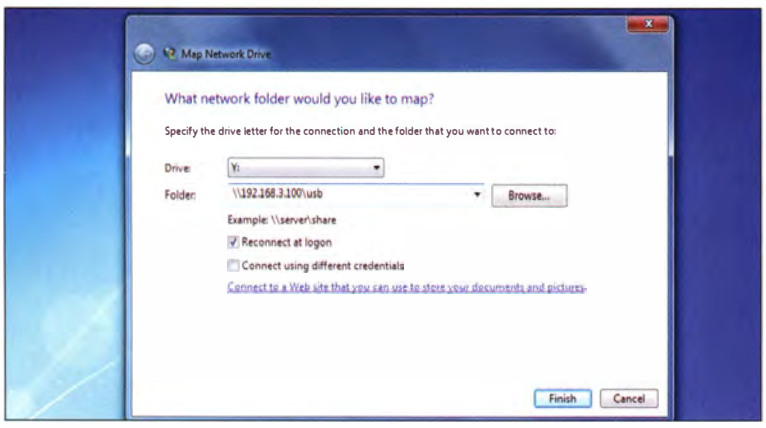
Torrent-Server:

Mayank Sharma zeigt Ihnen, wie Sie den Raspberry Pi als Torrent-Server nutzen und Dateien auf Ihr Smartphone streamen.



Der Raspberry Pi ist nicht nur ein toller Lerncomputer, sondern kann nebenbei auch noch als Server fungieren. Im Gegensatz zum weitverbreiteten Glauben braucht ein Server keine Unmengen an Rechenleistung. Um eine Datei herunterzuladen und im Netzwerk zu verteilen, ist beispielsweise keine Multicore-Maschine notwendig. Die Verwendung eines alten Linux-Computers als Server ist eine beliebte Art, ausrangierte Hardware einem neuen Zweck zuzuführen. Der Nachteil: Das verbraucht eine Menge Strom. Viele Modem-Router haben einen USB-Port und stellen die dort angeschlossene Hardware anderen Computern im Netzwerk zur Verfügung. Was aber, wenn Sie einen älteren Router haben oder mehr machen möchten, als bloß Dateien zu verteilen?

› **Einmal eingebunden, können Sie den mit dem Raspberry Pi verbundenen Speicher wie jedes lokale Laufwerk verwenden.**



Mit dem Raspberry Pi können Sie all das und mehr tun, ohne dabei die Betriebskosten eines normalen PCs zu haben, und gleichzeitig umgehen Sie die Einschränkungen eines Routers.

Die Zutaten

Wir verwenden einen Raspberry Pi Revision 2 mit 512 MB RAM, aber dieses Tutorial dürfte auch mit den älteren Versionen funktionieren. Das Betriebssystem ist Raspbian (Anleitungen zur Installation finden Sie auf Seite 16-21). Außerdem benötigen Sie eine Internetverbindung (siehe Seite 13-14 und Seite 69).

Nachdem Sie den Raspberry Pi angeschlossen und gestartet haben, begeben Sie sich auf die Admin-Seite Ihres Routers (wie genau, lesen Sie in der Beschreibung des Routers). Sehr wahrscheinlich wird Ihr Router DHCP nutzen, um die IP-Adressen an verbundene Hardware zu verteilen. Durchsuchen Sie das Admin-Interface und halten Sie nach einer Liste Ausschau, die alle angeschlossenen Geräte aufzählt. Später weisen Sie dem Raspberry Pi eine statische IP zu (siehe Kasten auf Seite 77), um sicherzustellen, dass er immer unter derselben Adresse erreichbar ist. Für die Zwecke dieses Tutorials nehmen wir zunächst an, dass dem Pi die dynamische Adresse **192.168.3.100** zugewiesen wurde. Mit dieser Information sind wir bereit, uns über SSH mit dem Pi zu verbinden. Jede Linux-Distribution wird mit einem SSH-Client ausgeliefert. Windows-Nutzer können auf das Tool *PuTTY* zurückgreifen. Um sich von einem Linux-Rechner aus zu verbinden, starten Sie ein Terminal und tippen folgenden Befehl ein:

```
$ sudo ssh pi@192.168.3.100
```

Nachdem Sie zugestimmt haben, diese Adresse den vertrauenswürdigen Hosts hinzuzufügen, fragt Sie das Fenster nach den Login-Daten. Auf einer frischen, nicht konfigurierten Raspbian-Installation ist „raspberry“ das Passwort für den Standardnutzer „pi“. Von diesem Punkt an sind alle Kommandos des Tutorials die gleichen, egal, ob Sie über einen Linux- oder Windows-Rechner mit Ihrem Raspberry Pi verbunden sind. Der Grund dafür ist, dass die Kommandos nicht lokal, sondern auf dem Pi ausgeführt werden. Da es sich um eine frische Installation handelt, werden Sie zu Beginn gefragt, ob Sie die Konfiguration durchführen wollen. Geben Sie folgenden Befehl ein:

```
$ sudo raspi-config
```

Das öffnet einen Bildschirm mit einer Vielzahl an Optionen. Scrollen Sie in der Liste bis ganz nach unten zum Punkt „Update“. Dieser lädt die aktuelle Version dieses Konfigurations-Werkzeugs herunter. Sobald das erledigt ist, wird der Raspberry Pi neu starten. Anschließend müssen Sie sich wie vorher mit dem **ssh**-Kommando oder per *PuTTY* neu verbinden. Das ist jedes Mal notwendig, wenn Sie irgendeine Einstellung verändern und der Pi neu starten muss.

Wählen Sie nun **expand_rootfs**, sobald Sie sich wieder im Einstellungsmenü befinden, damit Raspbian den kompletten Platz auf der SD-Karte verwenden kann. Nun ist die Option **memory_split** an der Reihe, die das RAM zwischen GPU und

Filesharing

CPU aufteilt. Da wir nur über eine Remote-Verbindung auf den Raspberry Pi zugreifen, sollten Sie der GPU die kleinstmögliche Menge von 16 MB zuweisen.

Nun sollten Sie auf dem Pi weitere Benutzer hinzufügen. Später werden wir den Zugang zu einigen Verzeichnissen auf angeschlossenen USB-Geräten für bestimmte Nutzer wie auch für eine Gruppe von Nutzern beschränken, wobei es trotzdem noch öffentliche Bereiche auf dem Gerät gibt.

```
$ sudo adduser bodhi
```

Dieser Befehl wird den Nutzer und die entsprechenden Verzeichnisse anlegen. Der Befehl fragt nach dem Passwort des Nutzers und anderen Details. Fügen Sie den User nun mit folgendem Befehl einer Gruppe hinzu:

```
$ sudo usermod -a -G users bodhi
```

Auf die Tanzfläche

Sobald Sie damit fertig sind, sollten Sie den Raspberry Pi für die Nutzer im Netzwerk zugänglich machen. Dazu verwenden wir *Samba*, das es uns erlaubt, Dateien über das CIFS-Protokoll („Common Internet File System") zugänglich zu machen. Verwenden Sie folgenden Befehl, um *Samba* auf dem Pi zu installieren:

```
$ sudo apt-get install samba samba-common-bin
```

Wenn das erledigt ist, müssen Sie in *Samba* noch die Nutzer hinzufügen. Beim Standardnutzer „pi" verläuft das folgendermaßen:

```
$ sudo smbpasswd -a pi
```

Sie werden dabei sofort nach einem Passwort gefragt. Normalerweise ist es sicher, auch hier das gleiche Passwort wie beim Benutzerkonto zu verwenden. Wiederholen Sie diesen Schritt für jeden Nutzer auf dem System. *Samba* wird über eine Konfigurationsdatei kontrolliert, die Sie vor der Benutzung anpassen müssen. Dabei ist es immer eine gute Idee, vor Änderungen ein Backup der Datei anzulegen:

```
$ sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.old
```

Nun verwenden wir den Kommandozeilen-Texteditor *Nano*, um die Konfigurationsdatei anzupassen:

```
$ sudo nano /etc/samba/smb.conf
```

Suchen Sie in dieser Datei nach dem Abschnitt „Authentification", und entfernen Sie die Raute vor der Zeile **#security = user**. So stellen Sie sicher, dass sich nur diejenigen über *Samba* verbinden können, die ein Home-Verzeichnis auf dem Raspberry Pi besitzen. Um den Nutzern Zugriff auf Ihre Home-Verzeichnisse zu gewähren, verändern Sie in der Konfigurationsdatei unter dem Abschnitt **[homes]** die beiden Optionen **browseable = yes** und **read only = no**. Drücken Sie STRG-X, um *Nano* zu beenden, speichern Sie dabei die Datei, indem Sie Enter drücken, sobald der Name der Datei angezeigt wird. Jedes Mal, wenn Sie irgendwelche Änderungen an Diensten vornehmen, egal welche, müssen Sie sie neu starten. Bei *Samba* geschieht das mit dem Befehl

```
$ sudo service samba restart
```

Nun ist es Zeit, die USB-Geräte in den Raspberry Pi einzustöpseln. Auch wenn der Raspberry Pi einige USB-Anschlüsse hat, ist es immer eine gute Idee, sämtliche USB-Hardware über einen USB-Hub mit eigener Stromversorgung zu verwenden.

Zu einer statischen IP wechseln

Da wir externe *Samba*-Freigaben automatisch einbinden, müssen wir sicher sein, dass sie immer unter der gleichen Adresse erreichbar sein werden. Mit aktiviertem DHCP wird der Router aber immer eine andere freie IP-Adresse verteilen. Für unseren Raspberry Pi, der aktuell unter **192.168.3.100** zu finden ist, bedeutet das, dass er bei einem Neustart eine neue IP zugewiesen bekommen könnte. In diesem Fall schlägt das Einbinden der Freigaben fehl. Um sicherzustellen, dass der Raspberry Pi immer die gleiche IP-Adresse erhält, müssen wir sie zu einer statischen IP-Adresse machen. Verschaffen Sie sich dazu erst Informationen über das Netzwerk, indem Sie den Befehl **ifconfig eth0** verwenden. Sie brauchen das Standard-Gateway sowie den Name-Server. In den meisten Fällen haben beide die gleiche Adresse, unter der Sie auch das Konfigurations-Interface des Routers finden. Die Netzmaske lautet fast immer **255.255.255.0**. Durchsuchen Sie die Konfigurationsseite des Routers, um den Bereich der DHCP-Adressen herauszufinden, die der Router verteilt. Sie möchten nämlich eine Adresse außerhalb dieses Bereiches, um zu verhindern, dass die Adresse einem anderen Gerät zugewiesen wird. Verbinden Sie sich nun über SSH mit dem Raspberry Pi, und ändern Sie die Datei **/etc/networks/interfaces.** Ersetzen Sie die Zeile

```
iface eth0 inet dhcp
```

durch

```
iface eth0 inet static
```

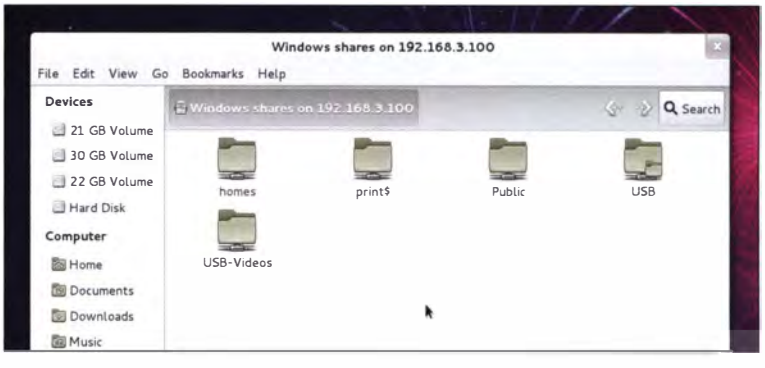
Unter diese Zeile schreiben Sie die Details zur statischen IP-Adresse:

```
address 192.168.3.121
netmask 255.255.255.0
gateway 192.168.3.1
nameserver 192.168.3.1
```

Das war's schon. Speichern Sie, und starten Sie den Raspberry Pi neu. Nach dem Neustart bestätigt die Eingabe von **ifconfig eth0**, dass die Verbindung die statische IP-Adresse hat, die Sie ihr zugewiesen haben. Verwenden Sie diese Adresse in allen Konfigurationsdateien.

Das gilt vor allem für größere USB-Massenspeicher, die viel Energie verbrauchen. Achten Sie auch darauf, wie das Laufwerk formatiert ist. Standardmäßig ist das bei vielen USB-Flashspeichern das Dateisystem FAT32. Das mag im Hinblick auf die Kompatibilität über viele Betriebssysteme hinweg das beste Dateisystem sein, allerdings ist es eines der schlechtesten, wenn es um das Verteilen von Dateien über ein Netzwerk geht. Dann gibt es noch NTFS, das von vielen externen USB-Laufwerken verwendet wird. Das allerdings ist ebenfalls kein geeignetes Format, wenn Sie Dateien von einem entfernten Laufwerk streamen möchten.

Mit **sudo fdisk -l** finden Sie den Pfad des Laufwerks heraus, »



› **Moderne Linux-Dateimanager wie *Nautilus* aus Gnome oder *Dolphin* von KDE können direkt auf *Samba*-Freigaben zugreifen und sie mounten.**

Einen Torrent erstellen

Der eine große Unterschied zwischen dem web- und dem desktopbasierten *Transmission* ist, dass ersteres keine grafischen Steuerelemente zum Erstellen ihrer eigenen Torrents besitzt. Für diesen Zweck müssten Sie das kommandozeilenbasierte Tool **transmission-create** verwenden, das mit **transmission-daemon** gebündelt ist. Falls Sie einen Torrent anlegen möchten, brauchen Sie nicht nur die Dateien, die Sie verteilen möchten, sondern auch einen BitTorrent-Tracker. Dabei handelt es sich um einen Server, der die Nutzer und Knoten verwaltet, welche den Torrent anbieten. Für dieses Tutorial verwenden wir den beliebten Tracker, der unter **http://linuxtracker.org:2710/announce** zu bekommen ist.

```
$ sudo transmission-create -o einlinux.torrent -t http://linuxtracker.org:2710/announce /pfad/zu/einerdistro.iso
```

Dieser Befehl erstellt einen Torrent mit dem Namen **einlinux.torrent**, der die Datei **einerdistro.iso** anbietet. Verweisen Sie beim Anlegen des Torrents statt auf die Datei auf ein Verzeichnis, um mehrere Dateien zu verteilen. Sobald Sie den Torrent erstellt haben, müssen Sie ihn nur an den *Transmission*-Client weitergeben, um die Verteilung der Inhalte mit anderen zu starten.

nachdem Sie es angesteckt haben. Der Befehl listet alle Speichergeräte auf sowie die Partitionen, die sich darauf befinden. Durchsuchen Sie die Ausgabe, und finden Sie das Laufwerk mit jener Größenangabe, die zu ihrem USB-Laufwerk passt. Die Bezeichnung des Laufwerks wird wahrscheinlich **sda** lauten, die der Partition **sda1**. Erstellen Sie nun einen Einhängepunkt und mounten sie das Laufwerk dort:

```
$ sudo mkdir /mnt/usb
$ sudo mount /dev/sda1 /mnt/usb
```

Das USB-Gerät bleibt eingehängt, bis Sie den Raspberry Pi neu booten. Um sich das ständige Remounten des Gerätes zu sparen, müssen Sie zuerst seine UUID herausfinden:

```
$ sudo blkid
/dev/sda1: LABEL="ntfs" UUID="3B5C053D35CAD865" TYPE="ntfs"
```

Fügen Sie diese zur Liste der beim Systemstart einzuhängenden Geräte hinzu:

```
$ sudo nano /etc/fstab
UUID=3B5C053D35CAD865 /mnt/usb ntfs defaults 0 0
```

Samba wurde darauf ausgelegt, diejenigen Dateien und Verzeichnisse zur Verfügung zu stellen, die in der Konfigurationsdatei festgelegt wurden. Wir gehen einmal davon aus, dass sich auf dem USB-Speicher einige Verzeichnisse befinden:

```
$ ls /mnt/usb
documents downloads music videos
```

Um den Ordner „downloads" im Netzwerk zur Verfügung zu stellen, öffnen Sie die Datei **/etc/samba/smb.conf** mit *Nano*. Scrollen Sie zum Ende, und fügen Sie folgende Zeilen hinzu:

```
[Downloads]
comment = Place all your downloads here
Path = /mnt/usb/downloads
browseable = yes
writable = yes
read only = no
valid users = @users
```

Dieser Textblock stellt das Verzeichnis **/mnt/usb/downloads** allen Nutzern der Gruppe **users** zur Verfügung. Später werden wir dieses Verzeichnis auf Windows- und Linux-Rechnern im Netzwerk einbinden. Die Anwender sind dann in der Lage, ihre Downloadmanager so zu konfigurieren, dass Dateien von überall aus dem Netzwerk direkt in dieses Verzeichnis auf dem mit dem Pi verbundenen USB-Gerät gespeichert werden können. Sie haben aber auch die Möglichkeit, einige Ordner nur bestimmten Nutzern zugänglich zu machen:

```
[Documents]
comment = Important eyes-only PDF files
path = /mnt/usb/documents
browseable = no
writable = yes
read only = no
valid users = pi, bodhi
```

In diesem Beispiel können nur die Nutzer „pi" und „bodhi" die Inhalte des Ordners **documents** zugreifen.

Den Torrent-Server aufsetzen

Torrents sind die bevorzugte Methode zum Weiterverbreiten von Open-Source-Inhalten. Die meisten Linux-Distributionen werden auf diese Weise verteilt, entweder über einen eigenen Tracker oder über **linuxtracker.org**. Unter Linux gibt es keinen Mangel an Torrent-Clients. Wir nutzen *Transmission* wegen seines einfach zu benutzenden Web-Interfaces. Wir installieren *Transmission* auf Raspbian, können ihn von überall aus aufrufen und Torrents hinzufügen, überwachen und steuern. Um *Transmission* zu installieren, geben sie über eine SSH-Verbindung zu Ihrem Raspberry Pi folgenden Befehl ein:

```
$ sudo apt-get install transmission-daemon
```

Das installiert und startet den *Transmission*-Dienst. Aber bevor Sie ihn zum Herunterladen von Torrents nutzen können, müssen Sie ihn noch einrichten. Stellen Sie sicher, dass *Transmission* nicht läuft, bevor Sie Änderungen an der Konfigurationsdatei vornehmen:

```
$ sudo service transmission-daemon stop
```

Fügen Sie, bevor Sie weitermachen, den User „transmission", der bei der Installation automatisch angelegt wird, Ihrer eigenen Benutzergruppe hinzu:

```
$ sudo mkdir /mnt/usb/public
$ sudo chown debian-tranmission /mnt/usb/public
```

In der *Samba*-Konfigurationsdatei nehmen Sie eine weitere Freigabe mit auf:

```
[Public]
comment= Public share for torrents
browseable = yes
path = /mnt/usb/public
public = yes
writeable = yes
guest ok = yes
```

Starten Sie *Samba* neu, um die Freigabe allen zugänglich zu machen. Nun ist es Zeit, den Dienst für *Transmission* zu konfigurieren. Die Einstellungen sind in der Datei **/etc/transmission-daemon/settings.json** festgelegt. Öffnen Sie die Datei in *Nano*, und ändern sie zu Beginn folgendes

```
„rpc-whitelist-enabled": true
```

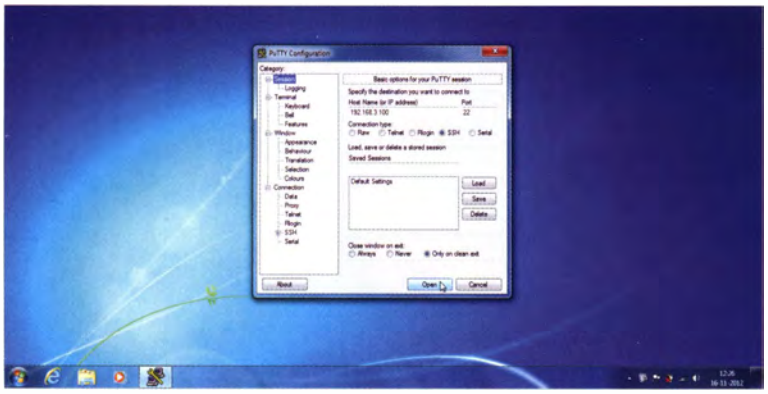
zu

```
„rpc-whitelist-enabled": false
```

damit man sich von allen Computern aus verbinden kann. Legen Sie nun das Downloadverzeichnis mit folgendem Befehl fest:

```
„download-dir": „/mnt/usb/public/downloads/Complete"
```

› Unter Windows verwenden Sie *PuTTY*, um sich per SSH mit dem Raspberry Pi zu verbinden. Geben Sie dazu die IP-Adresse des Pi ein und klicken sie auf „Open".



Sie können für unvollständig heruntergeladene Dateien auch einen anderen Ordner angeben. Ändern Sie dazu

„incomplete-dir-enabled": false

zu

„incomplete-dir-enabled": true

und geben Sie dann in der folgenden Zeile das Verzeichnis an, wo unvollständige Downloads gespeichert werden sollen:

„incomplete-dir": „/mnt/usb/public/downloads/Incomplete"

Benutzerauthentifizierung

Da wir den Ordner **/mnt/usb/public** dem Besitz des Benutzers „transmission" übergeben haben, wird dieser automatisch alle neuen Verzeichnisse darin anlegen. Während die heruntergeladenen Torrents öffentlich sind, können Sie festlegen, dass nicht jedermann neue Torrents zum Download einreihen kann. Die Authentifikation des Benutzers ist ein Weg, das zu erreichen. Vor der Anmeldung hat der Nutzer keinen Zugriff auf *Transmission*. Für diese Einstellung ändern Sie zunächst

„rpc-authentication-required": false

zu

„rpc-authentication-required": false

Dann geben Sie unter

„rpc-password": „<password>"

ein Passwort an, das *Transmission* automatisch verschlüsselt. Speichern Sie nun die Datei, und starten Sie den Dienst mit **sudo service transmission-daemon start** neu. Standardmäßig läuft *Transmission* über den Port 9091. In unserem Beispiel lautet die komplette URL für das Web-Interface für *Transmission* **192.168.3.100:9091**. Rufen Sie diese Adresse mit einem Browser auf, erscheint eine Passwortabfrage. Der Benutzername ist „transmission", das Passwort ist jenes, das Sie vorhin in der Konfigurationsdatei angegeben haben.

Bevor Sie einen Torrent herunterladen können, brauchen Sie zuerst die Adresse der Torrent-Datei. Klicken Sie im Web-Interface von *Transmission* auf den „Open Torrent"-Button. Fügen Sie im auftauchenden Fenster die Adresse zur Torrent-Datei ein, und klicken Sie auf „Upload", um den Download zu starten. Die Benutzeroberfläche ist sehr einfach zu verstehen. In der Standardeinstellung zeigt sie alle hinzugefügten Torrents an, Sie können aber auch die Drop-Down-Menüs verwenden, um die Torrents nach ihrem Status oder Tracker zu sortieren. Wenn Sie auf einen Torrent in der Liste rechtsklicken, wird ein Kontextmenü angezeigt.

Sobald ein Torrent fertig heruntergeladen wurde, wird er aufgrund unserer Konfiguration automatisch in den öffentlich zugänglichen Ordner **/mnt/usb/public/downloads/Complete** verschoben. Die einfachste Weise, um auf eine Freigabe unter Linux zuzugreifen, ist die Eingabe der Adresse in den Dateimanager. Die meisten modernen Dateimanager, so wie *Nautilus* aus Gnome, unterstützen *Samba*. Starten Sie *Nautilus*, und drücken Sie die Tastenkombination STRG+L um die Adressleiste aufzurufen. Geben Sie nun **smb://** ein, gefolgt von der IP-Adresse des Raspberry Pi, auf dem *Samba* läuft. In unserem Fall wäre das **smb://192.168.3.100**. Um auf eine bestimmte Freigabe zuzugreifen, können Sie deren Namen an das Ende der Adresse anhängen, beispielsweise **smb://192.168.3.100/documents**. Oder Sie hängen die Freigabe mittels folgendem Befehl in Ihr Dateisystem ein:

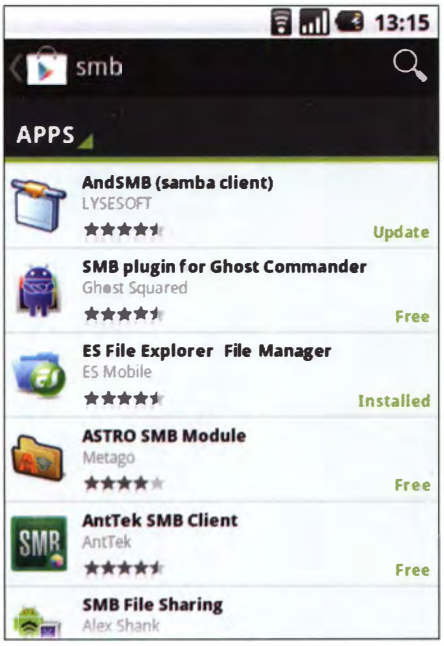
```
# mount -t cifs -o username=pi,password=raspberry
//192.168.3.100/usb/downloads /mnt/downloads
```

Um diese Freigabe bei jedem Start automatisch einzuhängen, können Sie folgende Zeile der Datei **/etc/fstab** hinzufügen:

```
//192.168.3.100/usb/downloads /mnt/downloads cifs
username=pi,password=raspberry 0 0
```

■

Mit Android auf Dateifreigaben zugreifen



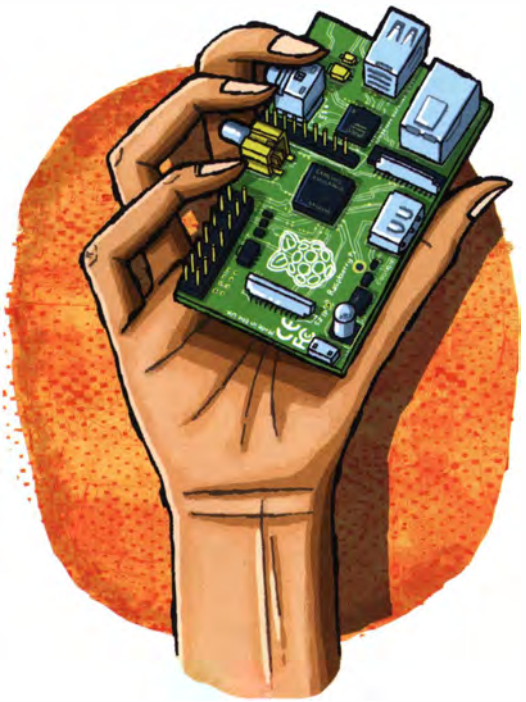
1 Suchen und Installieren
Der Android Play-Store ist voll mit Dateimanagern, die mit *Samba*-Freigaben umgehen können. Wir verwenden den beliebten *ES File Explorer File Manager*.

2 Konfigurieren
Stellen Sie in der App die Ansicht von Local auf LAN, öffnen Sie Menü > New > Server, und geben Sie die Verbindungs- und Anmeldedaten ein.

3 Durchsuchen und streamen
Abhängig von Ihren Rechten können Sie Dateien herunter- und hochladen, sie streamen sowie auf öffentliche wie private Bereiche der Freigabe zugreifen.

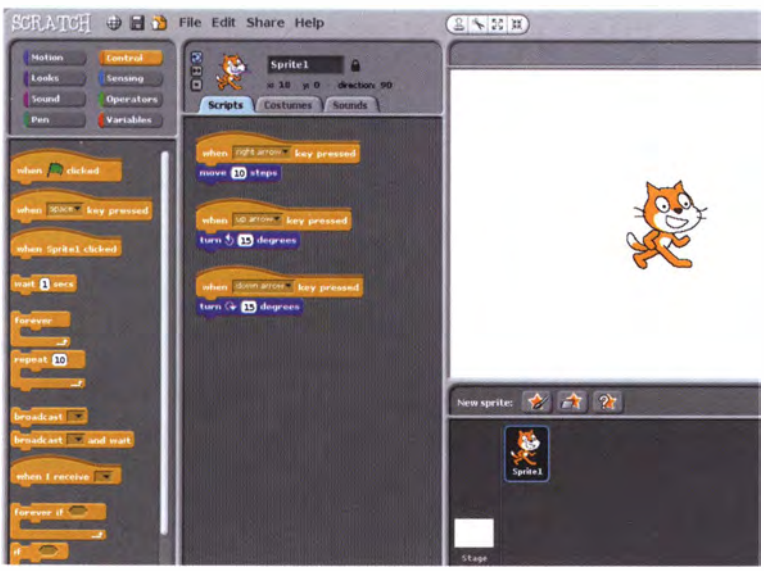
Programmieren:

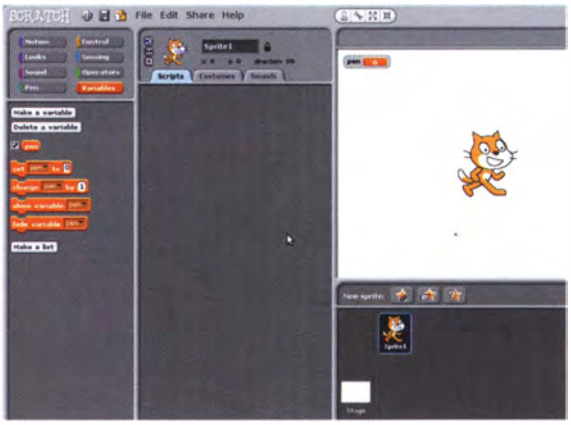
Ben Everard zeigt Ihnen, wie man die beiden Standardprogrammiersprachen für Raspbian in den Griff bekommt und wie man ein paar einfache Programme erstellt.



› **Abbildung 1: Da die Blöcke farblich kodiert sind, wissen Sie immer genau, von welchem Menü Sie auswählen können.**

In diesem Tutorial versuchen wir uns auf die Grundidee des Raspberry Pi zurückzubesinnen, nämlich, Einsteigern die Technik näherzubringen. Auf den folgenden Seiten bekommen Sie eine kleine, gut machbare Einführungsrunde in die beiden bei Raspbian mitgelieferten Programmiersprachen. Scratch, die erste der beiden Programmiersprachen, ist besonders gut dazu geeignet, sich Coding-Grundkenntnisse anzueignen. Da alle Befehle vorgefertigt sind und Sie sie per Drag-and-Drop in





› **Abbildung 2: Den Variablen wird per Voreinstellung eine „0“ zugeordnet. Im Malbereich können Sie immer deren aktuellen Wert im Auge behalten.**

Ihr Skript integrieren können, brauchen Sie sich auch keine Befehle zu merken. Wir beginnen mit einem einfachen Malprogramm, das Sie mithilfe der Pfeiltasten Linien auf Ihrem Bildschirm zeichnen lässt. Zuerst müssen wir einen Code schreiben, der uns die Katze bewegen lässt. Dafür verwenden wir drei verschiedene Blöcke, von denen jeder bei einem bestimmten Tastendruck ausgeführt werden soll.

Klicken Sie also auf **Steuerung**, und ziehen Sie den Block **Wenn Taste <Leertaste> gedrückt** in das Feld **Skripte**. Dadurch wird ein Skript erstellt, das sich immer dann aktiviert, wenn die Leertaste betätigt wird. Benutzen Sie nun das Dropdown-Menü, um **<Leertaste>** durch **<Pfeil nach rechts>** zu ersetzen, und ziehen Sie danach den Block **gehe <10>er-Schritt** in Ihr Skript direkt unter den vorherigen Befehl. Damit bewegen Sie die Katze vorwärts, sobald Sie die rechte Pfeiltaste drücken. Im nächsten Schritt erstellen Sie ähnliche Skripte, mit denen Sie die Katze im Uhrzeigersinn drehen, wenn die Pfeiltaste nach unten gedrückt, oder gegen den Uhrzeigersinn, wenn die Pfeiltaste nach oben gedrückt wird. Um einen Eindruck zu bekommen, wie das Ganze aussehen soll, schauen Sie sich einmal Abbildung 1 an. Da wir uns nun auf dem Bildschirm umherbewegen können, brauchen wir einen weiteren Block, der uns zeichnen lässt. Da wir nicht die ganze Zeit eine Linie ziehen wollen, nutzen wir die **Stift**-Funktionen von Scratch: Bei **senke Stift** ist der Stift aufs „Papier“ gedrückt und zeichnet, bei **hebe Stift** nicht.

Damit wir zwischen den beiden Zuständen des Stifts umschalten können, muss sich unser Programm jedoch merken, in welchem Zustand er sich gerade befindet. Hierfür verwenden Programmiersprachen Variablen, in denen kleine Informationen hinterlegt und ausgelesen werden können. Bevor Sie jedoch eine Variable verwenden können, müssen Sie ihr Speicher zur Verfügung stellen. Gleichzeitig geben wir ihr noch einen Namen, damit wir in den Befehlen auf sie verweisen können. Gehen Sie

Basics

dafür auf **Variable erstellen**, und geben Sie ihr einen Namen. Nun können Sie eine Auswahl Befehle sehen, die die Variable benutzen oder ihren Inhalt verändern können. Da wir jetzt Informationen speichern können, müssen wir dem Computer sagen, wie er sich im Bezug auf die Variable verhalten soll. Hierfür verwenden wir einen **falls <...> sonst <...>**-Block, der überprüft, ob eine Aussage wahr oder falsch ist. Ist sie wahr, wird der erste Block, andernfalls der zweite ausgeführt. In unserem Programm verwenden wir die Variable für **Stift**. Bei einer **0** wird der Stift aufgesetzt und der Zustand auf **1** umgestellt. Andernfalls heben wir den Stift an, und die Variable wird auf **0** gestellt. Mit dieser Methode können wir zwischen beidem wechseln, indem wir die Leertaste betätigen (siehe Abbildung 3). Beachten Sie den **=**-Operator in unserer **falls <...>**-Zeile. Dieser bewirkt, dass unser erster Code-Block nur ausgeführt wird, wenn unsere Variable **Stift** eine **0** enthält (**equals**), anderfalls (**else**) wird der zweite Block ausgeführt.

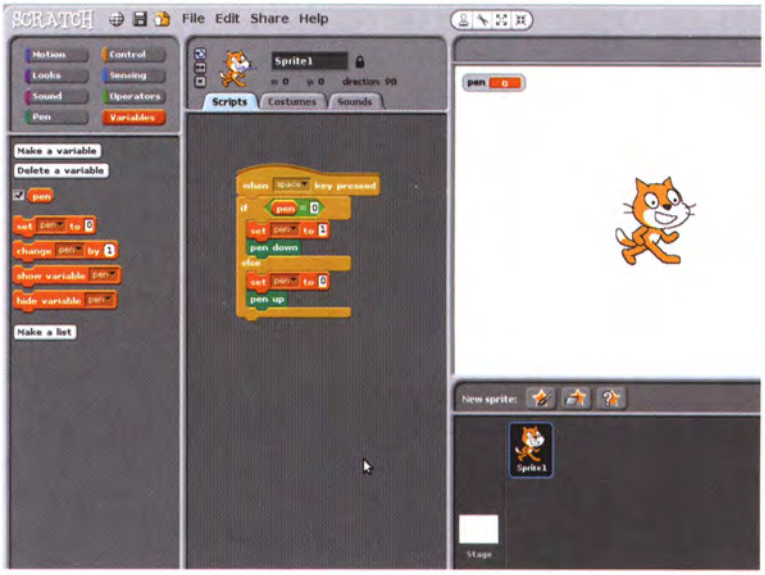
Programmschleifen

Prima, nun können Sie die Katze bewegen und ein Bild malen. Jetzt wollen wir mal einen Kreis versuchen. Es wird zwar eher eine geometrische Figur mit vierundzwanzig Seiten, aber das kann man gelten lassen. Unsere Methode wird sein, die beiden Befehle **gehe <10>er-Schritt** und **drehe dich <im Uhrzeigersinn> um <15> Grad** so lange zu wiederholen, bis ein kompletter Kreis geschlossen ist. Klar könnten Sie diese zwei Zeilen einfach vierundzwanzigmal wiederholen und hätten Ihr Ziel erreicht, aber das wäre nicht sonderlich sauber programmiert. Auch würde es eine Weile dauern, und wenn Sie die Größe Ihres Kreises verändern wollten, müssten Sie die ganze Prozedur wiederholen. Stattdessen werden wir eine Schleife benutzen, die sich immer wieder selber wiederholt. Es gibt eine Vielzahl verschiedener Schleifen. Einige setzen sich solange fort, bis eine Aussage falsch wird (vergleichbar mit einem **falls <...>**-Befehl, der durchgängig wiederholt wird), unsere jedoch wird eine im Vorfeld festgesetzte Anzahl von Kommandos abarbeiten. Das genaue Skript sehen Sie in Abbildung 4. Wie Sie sehen, kann Programmieren nicht nur Spaß machen, sondern Ihnen auch dabei behilflich sein, Ihren Rechner Aufgaben für Sie zu erledigen zu lassen. Mit Scratch sind hierbei Ihren Vorstellungen kaum Grenzen gesetzt. Sie könnten womöglich den nächsten Spielekracher, eine neue nützliche App oder sogar etwas derart Zukunftsweisendes erstellen, dass wir heutzutage noch nicht einmal einen Namen dafür haben. Zu Ihrer Inspiration finden Sie hier einige unserer persönlichen Lieblinge aus dem Internet (wenn Sie Flash installiert haben, können Sie die Programme einfach im Browser ausführen):

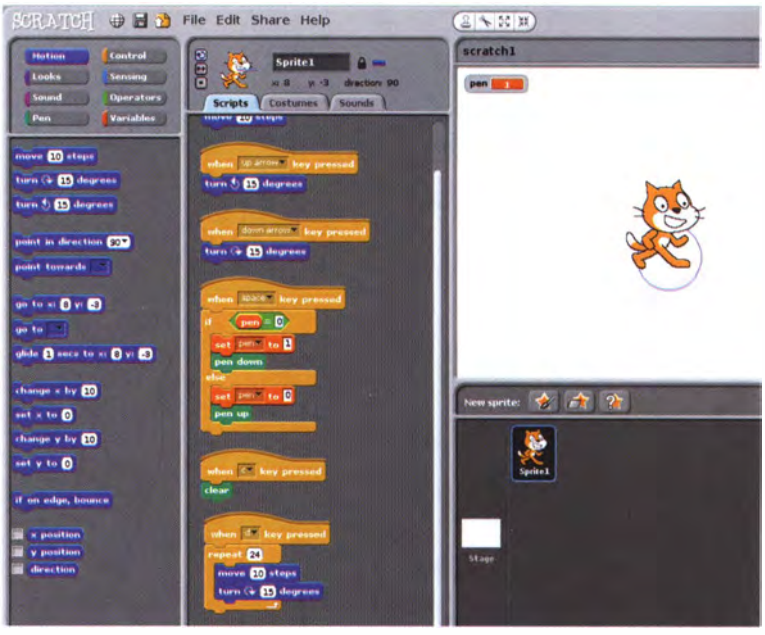
» **Super Mario Galaxy V4**: Erkunden Sie die Welt und sammeln Sie dabei Sterne ein.
http://scratch.mit.edu/projects/162167

» **Wipeout 2.0**: Basierend auf einer Fernsehshow, ist die Grafik nicht unbedingt modern, aber das Gameplay macht Spaß.
http://scratch.mit.edu/projects/1149306

» **Space War 4**: Ein klassischer Weltraumshooter.
http://scratch.mit.edu/projects/879463



› **Abbildung 3: falls <...> sonst <...>-Blöcke ermöglichen es Ihrem Rechner, Entscheidungen zu treffen, und sind somit der Grundbaustein jedweden Programmierens.**



› **Abbildung 4: Mit der gleichen Methode können Sie geometrische Figuren in Ihr Malprogramm integrieren.**

» **Snake Chamber**: Eine kleine Lehrstunde über Genetik und Schlangenzucht.
http://scratch.mit.edu/projects/2758178

» **Day Dream**: Scratch ist auch ein hervorragendes Werkzeug für Animationen.
http://scratch.mit.edu/projects/40150 »

Python

Um sich mit den Basics des Programmierens vertraut zu machen, ist Scratch sicherlich die erste Wahl. Früher oder später werden Sie jedoch an dessen Grenzen stoßen. Daher werden wir nun einen Blick auf Python werfen, eine beliebte universelle Programmiersprache.

Zuallererst sollten Sie wissen, dass Python – im Gegensatz zu Scratch – vollständig textbasiert ist. Das heißt natürlich nicht, dass Sie nicht auch Grafiken implementieren können, sondern dass Sie den Code selbst schreiben müssen, anstatt ihn per Drag-and-Drop zusammenzustellen.

Für die Erstellung eines Programms brauchen Sie folglich einen Texteditor. *Leafpad* wird bei dem Raspberry Pi mitgeliefert, daher werden wir einfach diesen verwenden. Sollten Sie jedoch Ihre Programmiererkarriere fortsetzen, wäre es durchaus ratsam, auf einen anderen Editor umzusteigen und ein wenig herumzuexperimentieren, um zu schauen, welcher Ihnen am besten liegt (*Geany* ist hierbei eine beliebte Alternative für Python-Neueinsteiger). Nebenbei bemerkt, Textverarbeitungsprogramme wie *LibreOffice Write* oder *Abiword* sind leider ungeeignet, da sie die Formatierung durcheinanderbringen.

Beginnen wir also, indem wir eine Datei in *Leafpad* öffnen und die erste Zeile schreiben:

```
#!/usr/bin/python
```

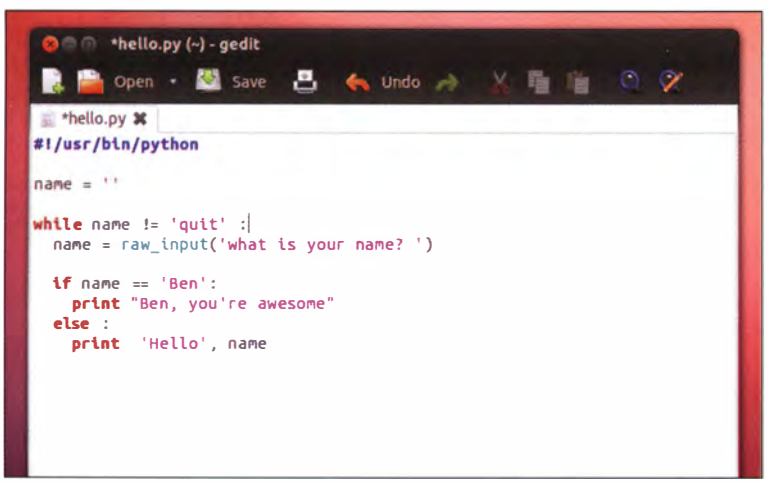
Diese Zeile, geheimnisvoll auch Shebang genannt, gibt dem System die Anweisung, das Programm *Python* im Dateiordner **/usr/bin/** zu starten. Dies wird bei all Ihren Python-Programmen benötigt.

Jetzt sind wir bereit, in die wunderbare Welt des Programmierens einzutauchen. Es ist Tradition in der Computerwelt, seine ersten Gehversuche mit der Textausgabe „Hello World!“ zu beginnen, und auch wir werden uns diesem Brauch nicht verschließen. Lassen Sie die zweite Zeile leer (das dient lediglich dazu, den Code besser lesbar zu halten), und tippen Sie in die dritte:

```
print "Hello World!"
```

Speichern Sie jetzt Ihr Werk in einer Datei mit dem Namen **hello.py**.

Um Ihr Programm auszuführen, müssen Sie mit einem Terminal an den Ort navigieren, an dem Sie die Datei abgespeichert haben. Hier müssen Sie dem System noch zu verstehen geben, dass es sich bei Ihrer Datei um ein ausführbares Programm handelt, indem Sie **chmod a+x hello.py** eintippen. Im Anschluss lässt sich Ihr Programm mit **./hello.py** starten. Jetzt sollte „Hello World!“ auf ihrem Bildschirm erscheinen.



Einige Texteditoren werden mit zusätzlichen Features fürs Programmieren ausgestattet. Das hier gezeigte *Gedit* hebt die Syntax Ihres Codes vor, um diesen lesbarer zu machen.

Module

Das beste Feature von Python ist sicherlich die schier unendliche Anzahl seiner Module. Mit diesen wird Python um zahlreiche nützliche Funktionen erweitert. Zwar werden nicht alle für den Anfänger geeignet sein, aber es ist sicherlich keine schlechte Idee, einen Eindruck davon zu bekommen, was alles mit der Programmiersprache möglich ist, sobald man sich ein wenig eingearbeitet hat. Dies sind unsere Top-5-Python-Module.

- **pyGames**: Spiele sind cool, Programmieren ist cool. Also muss auch Spiele programmieren cool sein. Dieses Modul hilft Ihnen dabei, Programme zu schreiben, mit denen sie Ihre karge Freizeit verplempern können.
- **pyGTK**: Früher oder später werden Sie womöglich auf die Idee kommen, Grafik-Apps zu programmieren. Dieses Modul wird Ihnen dabei helfen.
- **pyQT**: Für eine Gnome-Umgebung ist *GTK* hervorragend geeignet. KDE-Benutzer werden sich wahrscheinlich mit *QT* besser zurechtfinden.
- **RPi.GPIO**: Der Raspberry Pi hat eine große Auswahl an GPIO-Pins (siehe auch Seite 70-71), mittels derer Sie mit ihrer Umwelt interagieren können. Dieses Modul hilft ihnen dabei, diese anzusteuern.
- **NumPy:** Für Zahlenfetischisten! Manipulieren Sie Zahlen mit Leichtigkeit.

Dies zeigt uns, dass Ihr System einwandfrei funktioniert. Wie bei Scratch werden wir jetzt eine Interaktion mit dem User integrieren. Um dies zu bewerkstelligen, brauchen wir bei Python eine Variable, in der die Eingabe des Benutzers gespeichert wird. Löschen Sie die „Hello World!“-Zeile wieder, sodass nur die Shebang-Zeile übrig bleibt, und schreiben Sie stattdessen:

```
name = raw_input('What is your name? ')
```

Hiermit erschaffen Sie eine Variable mit der Bezeichnung **name**, lassen die Frage „What is your name?“ auf ihrem Bildschirm erscheinen und speichern die eingegebene Antwort des Benutzers in **name**. Da der Computer wissen muss, dass es sich hierbei um Text handelt, müssen wir diesen Bereich mit zwei Hochkommas begrenzen. Jetzt können wir diese Variable benutzen, um unseren **print**-Befehl ein wenig persönlicher zu gestalten:

```
print 'Hello', name
```

Da der Computer die Befehle in strikter Reihenfolge abarbeitet, muss diese Befehlszeile direkt unter die vorherige geschrieben werden. Vertauschen Sie die Reihenfolge, wird Ihr Programm einen Fehler ausgeben, da wir eine Variable benutzen würden, ohne sie vorher definiert zu haben. Speichern Sie nun Ihre Datei, und führen Sie sie mit **./hello.py** aus.

Entscheidungsmöglichkeiten

Auch wenn Ihr Programm hierdurch schon ein wenig an Funktionalität gewonnen hat, wirkt es trotzdem noch ein wenig leblos, da es immer die beiden gleichen Schritte abarbeitet und danach seinen Dienst einstellt. Um ein bisschen mehr Leben in die Sache zu bringen, brauchen wir eine Entscheidungs-

möglichkeit, bei der der Computer abhängig von der Eingabe des Users unterschiedlich reagiert.

Erinnern Sie sich noch an den **falls <...>**-Block von Scratch? Das gleiche geht natürlich auch in Python und heißt hier (wie auch in der Originalversion von Scratch) **if**. Eine rudimentäre **if**-Zeile würde folgendermaßen aussehen:

```
if <Aussage> :
<Einrückung>Befehlsblock
```

<Aussage> muss hierbei durch irgendetwas ersetzt werden, das entweder wahr oder falsch sein kann. In unserem Fall überprüfen wir einfach, ob der eingegebene Name zugeordnet werden kann.

```
if name == 'Ben' :
```

Sie stellen sich vermutlich die Frage: Warum **==** ? Computer (und in diesem Fall natürlich auch Programmierer) arbeiten nicht mit Mehrdeutigkeiten. Daher sollte jedes Symbol oder Wort, das wir benutzen, nur eine Bedeutung besitzen, andernfalls wird es kompliziert. Mit **=** wird einer Variable eine Wertigkeit zugeordnet, die wir mit etwas anderem vergleichen können. Wiederum müssen wir Ben mit Hochkommas umgeben, sodass der Computer weiß, dass es sich um Text handelt. Der Doppelpunkt zeigt dem Programm, dass hiermit die Zeile beendet ist und im Anschluss dem Rechner gesagt wird, was er damit zu tun hat.

Womöglich möchten wir, dass der **if**-Befehl mehr als eine Zeile Code enthält. Aus diesem Grund müssen wir unseren Programmier-Code in Blöcke gruppieren. Python verwendet in diesem Zusammenhang Einrückungen. Einrückungen können ein Leerzeichen oder ein Tabulator sein, jedoch ist es wichtig, dass Sie immer die gleiche Methode in ihrem Projekt verwenden, um die Übersicht zu behalten. Andernfalls kann es sehr schnell verwirrend werden, da Python sich nicht nach der Breite der einzelnen Einrückung richtet, sondern nach deren Anzahl.

Aber nun zurück zum Thema. Was wollen wir eigentlich machen, wenn **name == 'Ben'** ist? Nun, wir wollen ihn natürlich standesgemäß begrüßen:

```
if name == 'Ben' :
    print "Ben, you're awesome."
```

Beachten Sie die Einrückung am Anfang der zweiten Zeile und auch, wie wir Anführungszeichen verwenden. Das liegt daran, dass der Text, den wir einrahmen, bereits einen Apostroph enthält. Da wir zu anderen Benutzern nicht unhöflich sein wollen, fügen wir für den Fall, dass die **if**-Aussage falsch ist, noch einen weiteren **else**-Block hinzu.

```
else :
    print 'Hello', name
```

Ein letztes Feature, das wir noch implementieren, ist eine Schleife. Diese Programmschleife ist vergleichbar mit derjenigen, die wir schon in Scratch verwendet haben, jedoch wird diese nicht vierundzwanzigmal laufen. Stattdessen sagen wir ihr, wann sie aufzuhören hat. Hierfür verwenden wir die **while**-Schleife, deren Syntax folgendermaßen aussieht:

```
while <Aussage> :
<Einrückung>Befehlsblock
```

Das Programm soll sich beenden, wenn wir das Wort **quit** eingeben. Daher wird unsere Schleife so aussehen:

```
while name != 'quit' :
```

Problemlösung

Fragen Sie uns nicht warum, aber in vielen Programmiersprachen bedeutet ein Ausrufezeichen „nicht". Trotzdem stellt uns das vor einige Probleme. Setzen wir die Schleife vor **name = raw_input**, dann generiert das Programm einen Fehler, weil es nicht weiß, was **name** ist. Setzen wir sie jedoch darunter, wird uns einmal die Frage nach dem Namen gestellt und danach die Begrüßungsformel unendlich oft wiederholt werden.

Um diese Problematik zu umgehen, müssen wir vor **while** einen leeren String der Variable **name** zuordnen. Hiermit wird die Fehlerquelle behoben und die **while**-Aussage wird fehlerlos ausgeführt. Unser fertiges Programm sieht also nun so aus:

```
#!/usr/bin/python
name = ''
while name !='quit' :
    name = raw_input('What is your name? ')
    if name == 'Ben'
        print "Ben, you're awesome"
    else :
        print 'Hello', name
```

Beachten Sie bitte die längeren Einrückungen vor jeder **print**-Zeile. Das liegt daran, dass diese zweimal eingerückt wurden: einmal für die **while**-Schleife und ein weiteres Mal für die **if**-Aussage. Speichern sie nun ihr Programm als **hello.py** ab und führen Sie es als **./hello.py** aus. ■

Der interaktive Modus

Eigentlich benutzen wir Python, um Skripte auszuführen, aber es geht auch anders – mit dem interaktiven Modus. Auf diese Art und Weise geben wir eine Zeile ein und sie wird von Python umgehend bearbeitet. Danach geben wir eine weitere ein und so weiter und so fort. Wenn Sie schnell etwas ausprobieren möchten, ist dies ein durchaus nützliches Feature. Um die Python-Shell aufzurufen, geben Sie einfach **python** in die Kommandozeile ein. Ein **exit()** schließt Python wieder. Dieser Modus ist besonders nützlich, um mit neuen Bibliotheken zu experimentieren oder Einzelbefehle einzugeben, ähnlich wie bei *Bash*.

› **Python im interaktiven Modus**

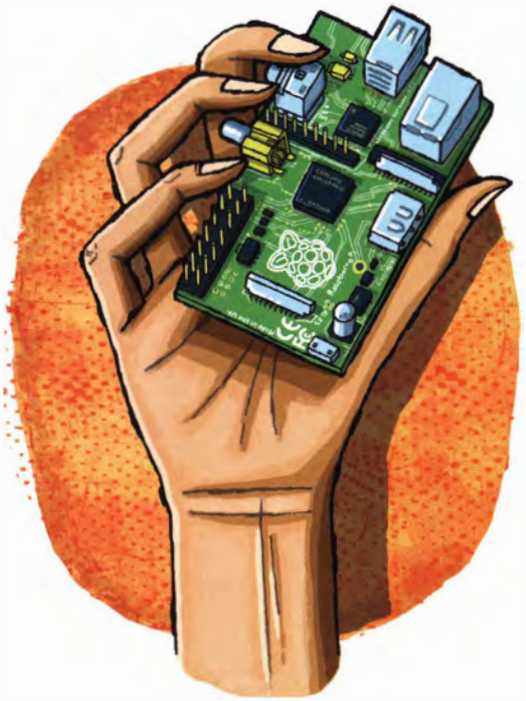
Verlassen Sie diesen Modus jedoch, werden all Ihre Mühen wieder gelöscht. Daher ist es eine bessere Idee, einen Texteditor zu verwenden, um Programme zu schreiben.

Wie weiter?

Wenn Ihnen unser Tutorial und das Schreiben Ihrer ersten Programme gefallen haben, werden Sie sich sicherlich fragen, was als Nächstes kommt. Scratch und Python sind bestimmt gute Programmiersprachen für Einsteiger. Suchen Sie sich also diejenige aus, die Ihnen am meisten gefallen hat. Sollten Sie sich für Scratch entschieden haben, werden Sie auf **http://scratch.mit.edu** fündig. Neben unzähligen Projekten gibt es hier Videotutorials und Einführungskurse, die Ihnen die Programmierumgebung näherbringen. Python hingegen ist weit mehr verbreitet, sodass Sie zum Lernen auf viel mehr Material zurückgreifen können. Auf die offiziellen Website **http://www.python.org** finden Sie ein gutes (wenn auch sehr trockenes) Tutorial. Zusätzlich gibt es zahlreiche ausgezeichnete Bücher zu dem Thema (eines davon wäre *Dive Into Python*, welches auch gratis auf der Seite **http://www.diveintopython.net** verfügbar ist.)

Entertainment:

Nach Jahren des Kampfes mit *MythTV* hat Graham Morrison dank *Tvheadend* eine bessere Lösung entdeckt.



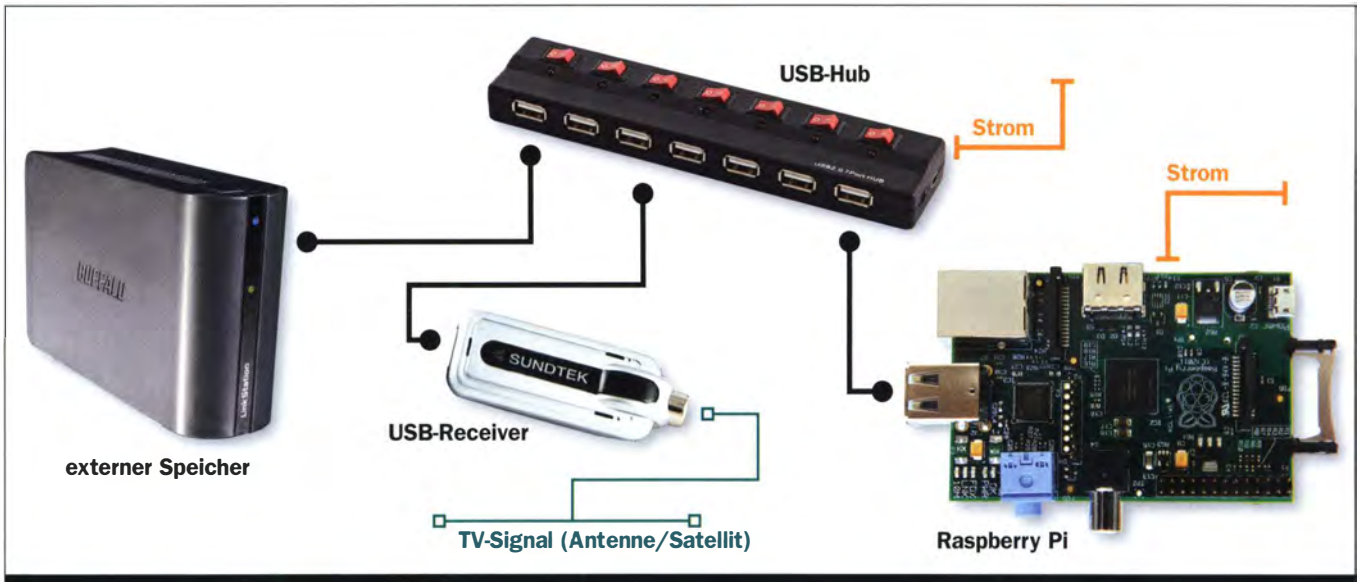
Viele der Set-Top-Boxen, die unter Fernsehern versteckt sind, basieren auf Linux. Trotz der schwachen Prozessorleistung sind sie dazu in der Lage, mehrere Kanäle gleichzeitig abzuspielen und zu speichern sowie Daten in Ihrem Heimnetzwerk zu streamen. Der Raspberry Pi eignet sich ebenfalls perfekt dazu und kann sich – mit der richtigen Hardware – in einen starken, kostengünstigen digitalen Videorekorder verwandeln, inklusive Streaming, Aufzeichnung und zeitversetztem Fernsehen.

Die richtige Hardware ist der Kernbegriff des vorherigen Absatzes: Ob die Installation problemlos verläuft, hängt meist davon ab, ob Ihre Hardware zur TV-Aufnahme fehlerfrei funktioniert. Zum Glück unterstützt Linux eine große Anzahl an Geräten, sodass diese ohne Modifikationen genutzt werden können.

Damit Sie alles gut nachvollziehen können, haben wir diesen Artikel in zehn Schritte aufgeteilt, die bei der Kommandozeile beginnen. Wenn Sie die Anleitung durchgearbeitet haben, werden Sie über ein vollständiges System zur Aufzeichnung digitalen Fernsehens verfügen, das mehrere Programme von verschiedenen Quellen mitschneiden kann. Alle Dateien werden dabei über unseren kleinen Raspberry Pi abgespielt. Der Pi ist das perfekte Back-End für das vor kurzem herausgekommene *XBMC*, das Sie als Front-End benutzen können – und zwar von jedem anderen Computer Ihres Netzwerks aus.

› ***XBMC* und *Tvheadend* bilden auf dem Raspberry Pi die perfekte Kombination.**

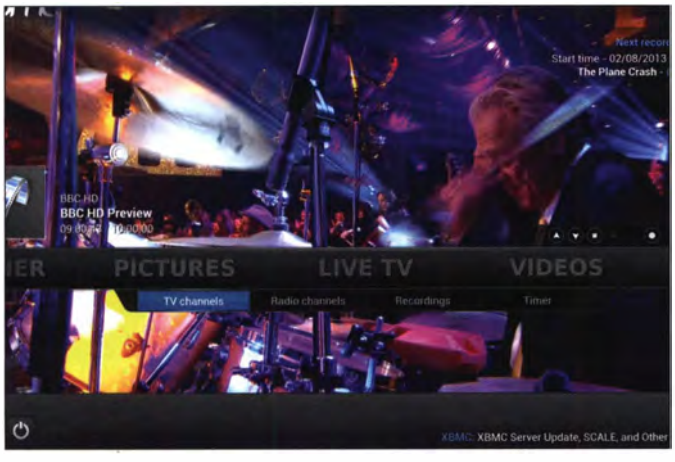
TV-Streaming



Schritt-für-Schritt-Anleitung

1 Unsere Hardware

Wir haben zwei USB-Receiver getestet und konfiguriert – den einen zum Empfang digitalen Fernsehens über eine Antenne, den anderen zum Empfang via Satellit. In diesem Artikel geben wir Ihnen Anweisungen zu beiden Empfangsarten. Für den DVB-T-Empfang (Antenne) haben wir den Sundtek MediaTV Pro und für den DVB-S-Empfang (Satellit) den Sundtek SkyTV Ultimate benutzt. Letzterer enthält einen 12-Volt-Adapter, den Sie auch anschließen müssen. Aber das Wichtigste ist: Schließen Sie die Geräte über einen USB-Hub mit externer Stromversorgung an den Raspberry Pi an. Wir haben zwei Tage mit erfolglosem Herumprobieren vergeudet, da wir zuerst keinen und danach einen inkompatiblen Hub verwendeten. In beiden Fällen schien zwar alles zu funktionieren, aber die Receiver haben einfach keine TV-Kanäle beim Scannen gefunden. Der Wechsel zu einem separat betriebenen, mit dem Raspberry Pi kompatiblen Hub hat das Problem gelöst, sodass wir diesen Punkt nicht oft genug betonen können. Schließen Sie den Hub an das Stromnetz an, danach den Receiver an den Hub und den Hub an den Raspberry Pi. Vergessen Sie nicht, die Antenne oder den Satellitenanschluss mit dem Receiver zu verbinden.



2 Externer Speicher

Wir gehen jetzt einmal davon aus, dass die aktuelle Version des Raspbian-Betriebssystems auf Ihrem Raspberry Pi installiert ist, dass der Pi mit dem Internet verbunden ist und dass Sie Ihre Befehle entweder direkt in die Konsole oder per SSH eintippen. Unsere nächste Überlegung: Wo speichern Sie Ihre Fernsehaufzeichnungen? Wir empfehlen Ihnen, eine externe Festplatte anzuschließen, da der permanente Lese- und Schreibzugriff die SD-Karte an Ihre Grenzen bringen wird. Um auf diese Weise Speicher hinzuzufügen, schließen Sie die Festplatte einfach an einen freien USB-Port an und überprüfen Sie die Ausgabe, indem Sie **tail /var/log/messages** eintippen. Danach müsste dort so etwas wie **usb 1-1.3.4.2: New USB device** stehen. Sie müssen noch den Bezeichner überprüfen: Hier müssten Sie etwas wie **sda: sda1** finden. Dabei ist **sda** das Gerät an sich und **sda1** die Partition. Tippen Sie **sudo mkdir /mnt/storage** ein, um einen Einhängepunkt einzustellen, und **sudo mount /dev/sda1 /mnt/storage/**, um Ihre externe Festplatte damit zu verbinden.

3 Treiber installieren

Abhängig von der Fernsehhardware, die Sie benutzen, könnte dieser Schritt überflüssig sein. Wenn Sie ein Gerät gewählt haben, das mit Linux kompatibel ist und keine zusätzlichen Treiber benötigt, dann schließen Sie es einfach an und machen Sie weiter mit Schritt 4. Wir mussten für unsere Sundtek-Geräte jedoch Treiber herunterladen und installieren. Das ist immerhin einfach – tippen Sie in die Kommandozeile:

```
wget http://www.sundtek.de/media/sundtek_netinst.sh
chmod 777 sundtek_netinst.sh
sudo ./sundtek_netinst.sh
```

Die letzte Zeile führt das Skript aus, das mittels der ersten Befehlszeile heruntergeladen wurde. Dieses ermittelt Ihr System und installiert die neueste Treiberversion. Das Skript lässt die Treiber laufen und konfiguriert sie so, dass sie beim Booten starten. Falls Sie DVB-T benutzen, müssen Sie zusätzlich folgenden Befehl ausführen:

```
/opt/bin/mediaclient --setdtvmode=DVBT
```

Dadurch stellen Sie sicher, dass die Karte für terrestrischen Empfang und nicht für den Empfang über Kabel konfiguriert ist, wozu das Gerät ebenfalls in der Lage ist.

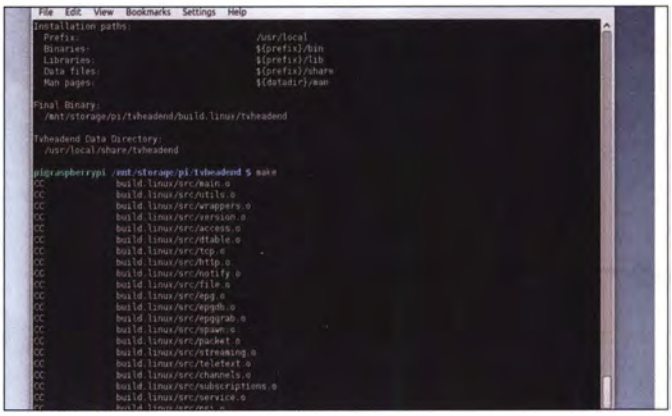
4 Tvheadend installieren

Mithilfe eines Plugins wird der Media-Player *XBMC* zu einem voll funktionsfähigen Videorekorder. Die Software, die wir für Aufzeichnung und Streaming des digitalen Fernsehens benutzen, nennt sich *Tvheadend*. Sie wird im Hintergrund die harte Arbeit erledigen – und zwar vom Raspberry Pi aus. Da *Tvheadend* ständig weiterentwickelt wird, haben wir die Entwicklungsversion benutzt. Sie können allerdings genauso gut die OpenElec-Distribution anstelle von Raspbian nutzen, wenn Sie Schritt 4 überspringen möchten. Zum Glück ist die Installation aber einfach. Zunächst benötigen Sie die Entwicklungs- und DVB-Tools:

```
sudo apt-get update
sudo apt-get install unzip libcurl4-openssl-dev pkg-config git
build-essential dvb-apps gcc-4.7
```

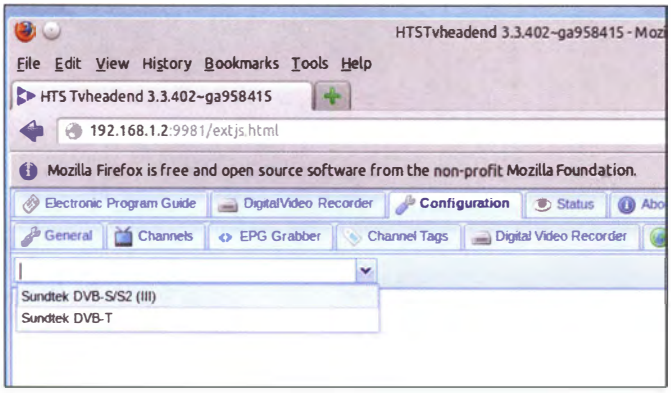
Zu guter Letzt laden Sie mithilfe von Git die neueste Tvheadend-Version vom Repository der Entwickler herunter und schließen den Erstellungsprozess mit dem folgenden, auf **./configure, make** und **sudo make install** basierenden Kommando ab:

```
CC=gcc-4.7 ./configure; make; sudo make install
```



5 Tvheadend konfigurieren

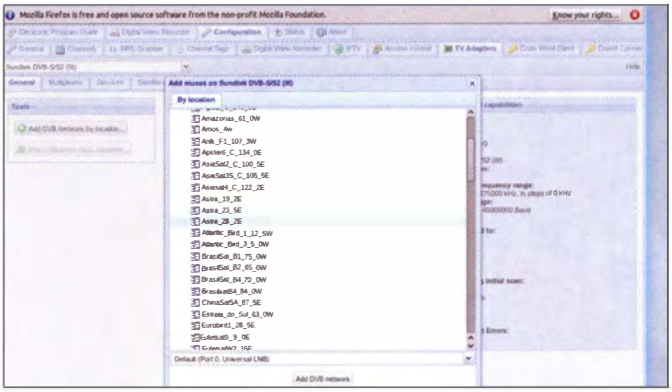
Sie greifen auf die Benutzeroberfläche von *Tvheadend* über den Webbrowser zu. Doch zuerst müssen Sie das Programm starten. Da dies das erste Mal ist, werden wir es im Konfigurationsmodus und als Daemon ausführen. Letzteres bedeutet, dass das Programm im Hintergrund läuft. Geben Sie **tvheadend -C -d** ein, öffnen Sie dann einen Browser (vorzugsweise auf einem anderen Rechner im selben Netzwerk), und tippen Sie Folgendes **http://<ip-adresse>:9981/extjs.html** ein. Die IP-Adresse Ihres Raspberry Pi finden Sie heraus, indem Sie **ifconfig** eingeben und nachsehen, was neben dem Feld „inet addr“ als Wert für das Gerät „eth1“ eingetragen ist. Ihr Browser wird das Standard-Front-End zu *Tvheadend* laden. Hier werden Sie alle Programmdaten sehen und Aufzeichnungen einstellen und anschauen. Sie müssen *Tvheadend* zuerst mitteilen, wie es die angeschlossene Hardware benutzen soll. Klicken Sie dazu auf „Configuration“ > „TV Adaptors“, und wählen Sie Ihr Gerät aus dem Drop-Down-Menü auf der linken Seite aus. Die Einstellungen, die Sie vornehmen müssen, unterscheiden sich etwas, je nachdem, ob Sie ein Satellitensignal empfangen oder eine Antenne benutzen. Daher teilen wir die nächsten Schritte auf. Wenn Sie einen Satellitenreceiver haben, fahren Sie mit Schritt 6 fort, bei Antennenempfang springen Sie direkt zu Schritt 7.



6 Satellitenempfang

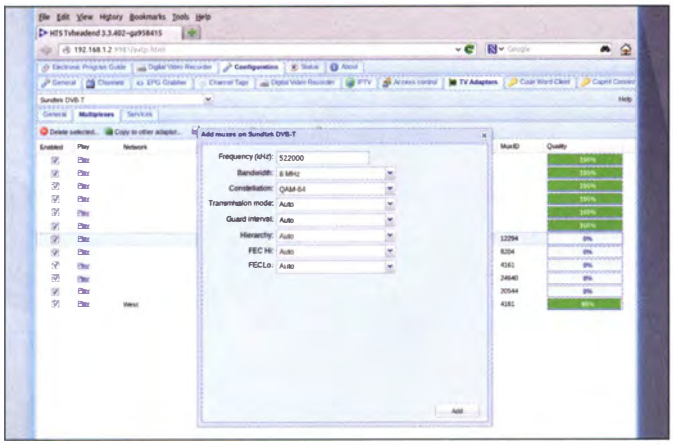
Ein Gerät mit Satellitenempfang ist am leichtesten zu konfigurieren. Nachdem Sie Ihren Adapter ausgewählt haben, sehen Sie im Reiter „General“ (Allgemein) einen Überblick der Konfigurationsmöglichkeiten. Klicken Sie zuerst auf „Enable“, dann auf „Save“. Wir wollen Kanalinformationen hinzufügen – dazu müssen wir die Daten eingeben, um einen Satelliten und dessen Bouquets (Multiplexe) zu finden, und anschließend diese Bouquets nach Kanälen durchsuchen. *Tvheadend* bündelt die Daten von Satellitenpositionen, sodass Sie nur auf die Schaltfläche „Add DVB Network by location“ (DVB-Netzwerk über Position hinzufügen) klicken müssen. Dadurch öffnen Sie ein Fenster mit einer Liste von Satelliten.

Nachdem Sie den Satelliten gewählt haben, wird *Tvheadend* eine Liste von dazugehörigen Bouquets zum Reiter „Multiplexes“ hinzufügen. Diese Liste wird nun nach Kanälen durchforstet – Sie können den Prozess im Kasten „Capabilities“ (Ressourcen) auf der rechten Seite mitverfolgen. Wenn alles funktioniert, müssten Sie hier viele Sender beziehungsweise Kanäle sehen, die zum Reiter „Services“ hinzugefügt werden. Falls Sie Ihr System nicht für den Antennenempfang konfigurieren wollen, fahren Sie mit Schritt 8 fort.



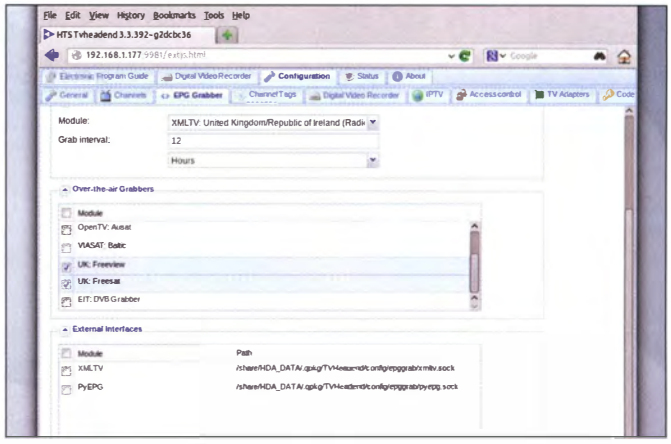
7 Antennenempfang

Als Erstes schalten Sie den Receiver im Reiter „General“ (Allgemein) ein. *Tvheadend* verfügt über eine umfassende Sendeanlagen-Datenbank, vorkonfiguriert mit den Daten für jedes Bouquet. Sie müssen wissen, auf welche Sendeanlage Ihre Antenne gerichtet ist. Hierbei könnte Ihnen für Deutschland der Link **www.ueberallfernsehen.de/dvbtdownloads127.pdf** helfen. Mit dieser Information klicken Sie einfach auf die „Add DVB Network“-Schaltfläche (DVB-Netzwerk hinzufügen) im Reiter „General“ und finden die Anlage über „By location“ (über Position). Falls Ihre Sendeanlage nicht aufgelistet ist und Sie die Details zu den Bouquets manuell eingeben müssen, klicken Sie im Reiter „Multiplexes“ auf „Add mux(es) manually“ (Bouquets manuell hinzufügen). Im sich daraufhin öffnenden Fenster müssen Sie die Frequenz, Bandbreite und Konstellation jedes Bouquets eingeben und alles andere auf „Auto“ stellen. *Tvheadend* wird die Multiplexes nach Sendern/Kanälen scannen und sie zu Ihrer Konfiguration hinzufügen.



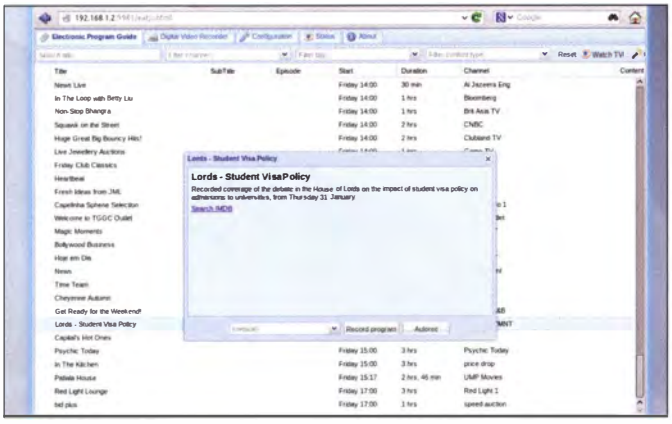
8 Kanäle hinzufügen

Nach dem letzten Schritt verfügen Sie hoffentlich über eine Reihe Sender. Nun müssen Sie *Tvheadend* erlauben, TV-Kanäle darin aufzuspüren, indem Sie im Reiter „General“ auf „Map DVB Services To Channels“ klicken. Im Idealfall verfügen Sie danach über eine Liste an Kanälen im Reiter „Channels“ außerhalb des Konfigurationsbereichs. Sie müssen die Angabe ändern, wo Aufzeichnungen standardmäßig gespeichert werden sollen – vermutlich zu dem Einhängepunkt, den Sie anfangs erstellt haben. Die Angabe ändern Sie, indem Sie im Reiter „Digital Video Recorder“ den Pfad für die Systemaufzeichnung („System Recording“) ändern. Um Änderungen wirksam zu machen, müssen Sie die Konfiguration des Reiters speichern. Außerdem müssen Sie den elektronischen Programmführer (EPG) konfigurieren. Normalerweise verfügt jeder Kanal nur über ein einfaches „Was läuft gerade?“ oder „Was kommt als Nächstes?“, aber manche Anbieter übermitteln eine umfangreiche 7-Tage-Programmvorschau. Um diese einzuschalten, müssen Sie im Reiter „EPG Grabber“ die betreffenden Sender im Untermenü „Over-The-Air-Grabbers“ per Häkchen aktivieren. Vergessen Sie nicht, zu speichern.



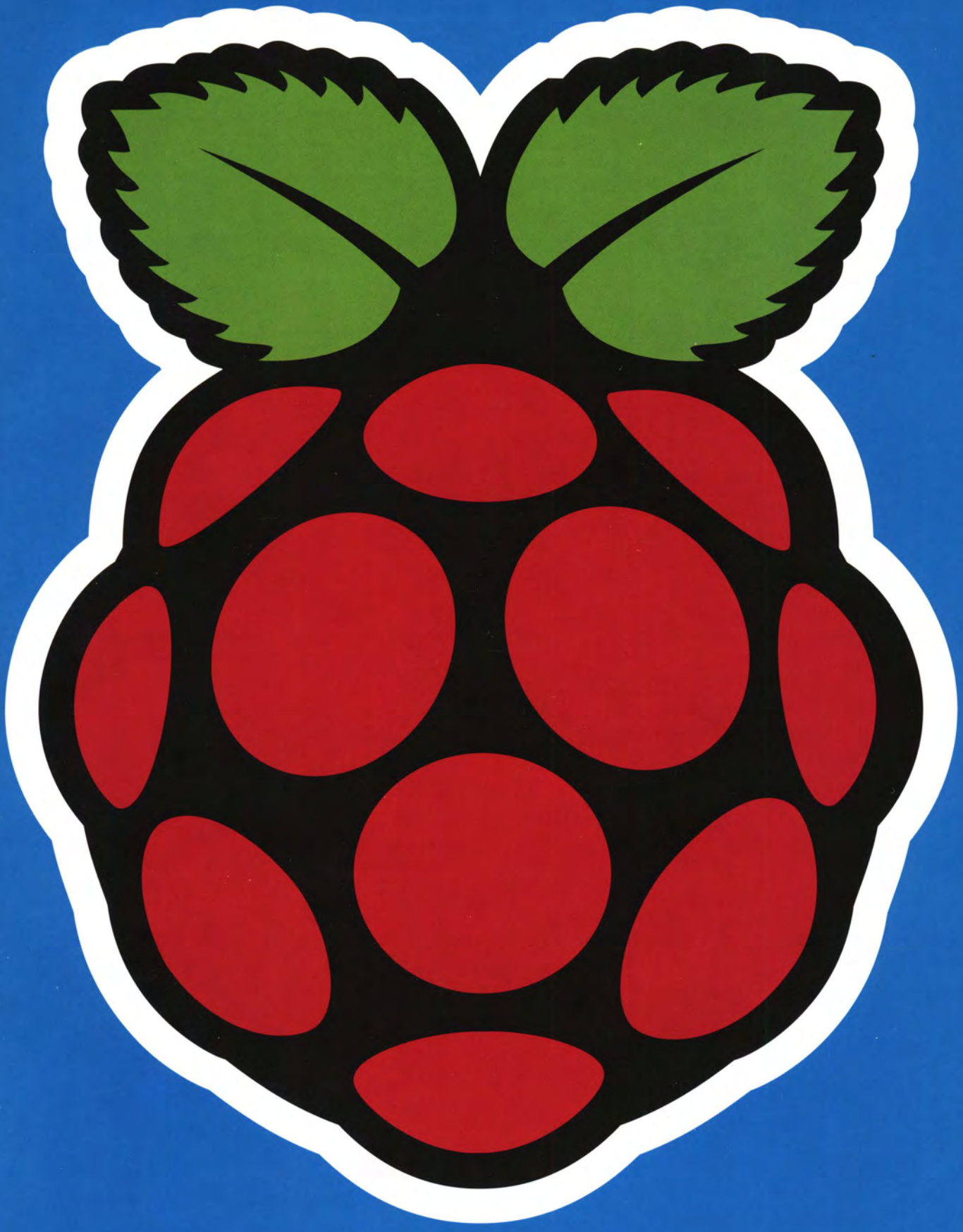
9 Ein Programm aufzeichnen

Der EPG-Reiter enthält Sendungen, die Sie anschauen oder aufnehmen können. Über einen Klick auf das jeweilige Programm öffnen Sie ein neues Fenster, wo Sie Aufzeichnungen konfigurieren können. Die Option namens Autorec ist aber interessanter: Sie richtet eine Suche ein, basierend auf den Programmdaten, sodass sie ganze Serien aufnehmen können, ohne sich darauf verlassen zu müssen, dass die Daten im EPG hinterlegt sind. Abhängig von der Menge an Kanälen und dem Umfang an EPG-Daten kann die Ansicht in diesem Reiter schnell unübersichtlich werden. Sie können jedoch mithilfe der Optionen zu Beginn der Liste die Ergebnisse filtern (etwa nach Kanal, Titel oder Schlagwort). Wenn Sie einen Filter besonders oft benutzen, klicken Sie auf die Schaltfläche „Create Autorec“ – dadurch fügen Sie die Suche zu *Tvheadend* hinzu, und das Programm wird alles aufzeichnen, was zum Suchbegriff passt. Um geplante Aufnahmen zu entfernen, klicken Sie auf „Digital Video Recorder“. Kommende Aufzeichnungen können über den ersten Reiter entfernt werden, während Autorec-Suchen über den letzten Reiter gelöscht werden. Die Reiter dazwischen nutzen Sie, um vorhandene Aufzeichnungen abzuspielen oder zu löschen.



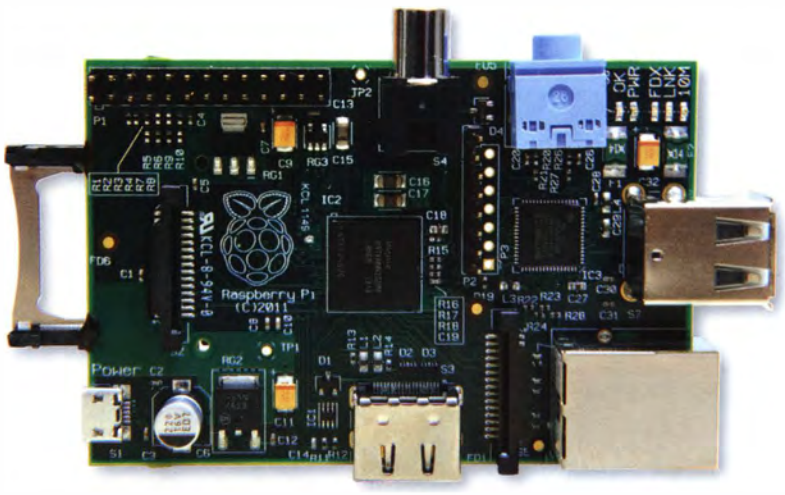
10 Aufzeichnungen und Live-Fernsehen schauen

Indem Sie ein *VLC*-Plugin für Ihren Browser installieren, können Sie Aufnahmen sowie Live-Fernsehen im Browser ansehen. Wir haben dieses Feature im *Firefox* getestet: Wenn Sie ein Programm anklicken, das gesendet wird, sehen Sie die Option, dieses abzuspielen („Play“). Falls Sie das *VLC*-Plugin nicht installiert haben, werden Sie gefragt, ob Sie dies tun wollen. Nach der Installation des Zusatzmoduls erscheint ein im Browser eingebettetes Fenster, das die Sendung zeigt. Über die Kontrollleiste am oberen Rand des Fensters können Sie auf Vollbild wechseln oder die aktuelle Wiedergabe pausieren. Auf die gleiche Weise können Sie bereits aufgezeichnete Programme über den „Digital Video Recorder“-Reiter aufrufen. Wenn Sie lieber auf den Browser verzichten, ziehen Sie per Drag-and-Drop die Netzwerk-URL in Ihren *VLC* oder einen anderen Player. Allerdings ist unserer Meinung nach die Kombination aus *XBMC* und *Tvheadend* die beste Möglichkeit, Letzteres zu benutzen. *XBMC* verfügt über ein Plugin, das direkt mit *Tvheadend* kommuniziert, den elektronischen Programmführer von Ihrem Pi herunterlädt und Ihnen erlaubt, Live-Kanäle anzuschauen sowie Sendungen aufzunehmen und wiederzugeben. Es ist zudem leicht zu konfigurieren: Besuchen Sie einfach die PVR-Plugin-Seite von *XBMC*. ■



Programmieren

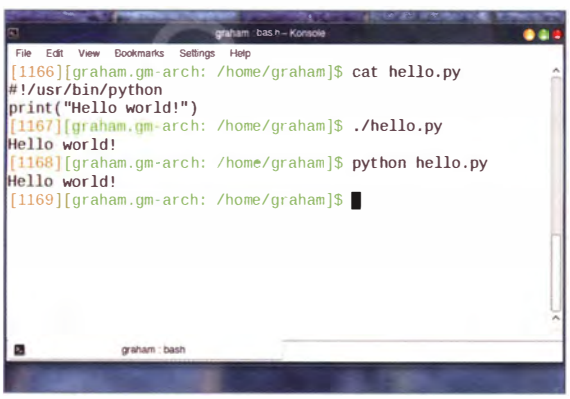
Die Schöpfer des Raspberry Pi hatten bei ihrem Projekt im Sinn, dass in Schulen viel mehr mit Computern gearbeitet werden solle, insbesondere im Hinblick auf das Programmieren. In diesem Teil des Handbuchs dreht sich alles um die Erstellung einfacher (und etwas komplizierterer) Skripte und Software mithilfe der Programmiersprache Python.



Python: So fangen

Alle Abenteuer mit dem Raspberry Pi starten mit einer einzelnen Codezeile. Und genau damit wollen wir hier beginnen.

Der Raspberry Pi wurde hauptsächlich als Bildungsinstrument entwickelt. Aber das wichtigste pädagogische Ziel war nicht das Betriebssystem oder die Hardware, sondern die Produktion einer billigen und leicht zugänglichen Plattform für die Programmierung. Die Pi-Erfinder liebten an den Heimcomputern der 1980er Jahre, dass man sie einschalten und sofort Code eingeben konnte, ohne vorher irgendetwas installieren zu müssen. Man konnte ohne Umschweife mit dem System herumexperimentieren und dadurch viel über das Programmieren und die Funktionsweise des Computers lernen. Wenn Sie Zugriff auf das Innerste Ihrer Maschine erhalten möchten, stellt Programmieren die beste Methode dar.



› **Python-Programme können Sie auf unterschiedliche Weisen laufen lassen: direkt vom Interpreter, durch Aufruf der Python-Anwendung oder durch Hinzufügen eines Shebang zu einer Skriptdatei.**

Moderne Computer arbeiten nicht mehr in der gleichen Weise. Hier haben Betriebssysteme und Befehlszeilen die BASIC-Interpreter der alten Maschinen ersetzt. Aber Programmierung ist auch heute noch der beste Weg, etwas über Ihren Rechner zu erfahren, und was vielleicht noch wichtiger ist, es ist eine der besten Möglichkeiten, Kompetenzen zu erlangen, die auf fast alle Plattformen, Systeme und Programmiersprachen übertragen werden können. Außerdem benötigen Sie keine besondere Erfahrung, um anzufangen. Weil die meisten Anweisungen, die wir verwenden, über die Befehlszeile eingegeben werden, ist es sinnvoll, sich als Erstes mit dieser vertraut zu machen. Es hilft, wenn Sie wissen, wie Programmiersprachen arbeiten, aber selbst dann ist der beste Weg zu lernen, ins kalte Wasser zu springen.

Die offizielle Programmiersprache des Raspberry Pi heißt Python. Dies ist eine fest etablierte Sprache, die bereits von Millionen Entwicklern benutzt wird. Python wurde für den Raspberry Pi in erster Linie deshalb ausgewählt, weil es einfach zu lernen und ideal zum Experimentieren geeignet ist. Aber diese Programmiersprache ist auch eine mächtiges Werkzeug, das für viele ambitionierte Projekte bei Google und vielen neuen Technologien eingesetzt wird. Python ist überdies plattformunabhängig und beinhaltet die gleichen Konzepte und Paradigmen wie fast jede andere Programmiersprache. Wenn Sie sich also näher mit Python beschäftigen, wird Ihnen dies nicht nur helfen, das Maximum aus Ihrem Raspberry Pi herauszuholen, sondern es kann Ihnen auch die Welt des Computers eröffnen, so wie das 1983 die alten 8-Bit-Maschinen taten.

Python-Grundlagen

Python ist bereits auf Ihrem Raspberry Pi installiert, aber Sie können es auch auf jeder anderen Maschine, die Sie besitzen, installieren. Es gibt Versionen für Windows, OS X und Linux sowie für weitere Systeme. Sie können testen, ob Python bereits installiert ist, indem Sie ein Terminalfenster öffnen und das Wort **python** eingeben. Wenn alles funktioniert, sehen Sie ein paar Zeilen Ausgabe, die mit der Zeichenkette >>> beginnen. Es handelt sich um den Python-Interpreter und damit um das, was Python zu einer solch guten Sprache macht. Vielleicht wissen Sie noch nicht, was ein Interpreter überhaupt ist, doch das ist schnell erklärt: Alles, was er im Grunde macht, ist, Ihre Eingaben zu übersetzen und ihre eingegebenen Wörter in Aktionen umzuwandeln. Geben Sie zum Beispiel **print("Hello world!")** ein. Sobald sie die Enter-Taste gedrückt haben, wird der Python-Interpreter Ihre Eingabe decodieren und entsprechend Ihrem Kommando den Text „Hello world!“ anzeigen.

Allerdings besitzen die meisten Programmiersprachen keinen Interpreter mehr. Stattdessen schreiben Sie die Anweisungen in eine Datei und benutzten ein anderes Kommando, um Ihren Code in die Aktionen umzusetzen, die Sie benötigen. Mit Python können Sie dies ebenfalls tun. Geben Sie **quit()** ein, um den Interpreter zu beenden, und öffnen Sie anschließend einen Texteditor. Einen einfach zu bedienenden Editor namens Nano

Sie an

können Sie aus der Kommandozeile heraus aufrufen. Wenn Sie beispielsweise **nano hello.py** eingeben, wird Nano geöffnet, eine Textdatei mit dem Namen hello.py kreiert und der Cursor an oberster Stelle in der Datei platziert, sodass Sie direkt mit der Eingabe beginnen können. Schreiben Sie nun wieder **print("Hello world!")** und drücken Sie Steuerung + X. Sie werden gefragt, ob Sie die Datei speichern wollen, und beantworten dies mit Y. Sie haben damit eine neue Quelldatei erstellt, deren Inhalt Sie mit der Eingabe **python hello.py** ausführen können.

Die Endung **.py** der Datei ist nur dazu da, dass Sie wissen, dass es sich um Python-Code handelt, und wird sonst nicht von Linux verwendet. Wenn Sie der Kommandozeile mitteilen möchten, dass es sich um Python-Code handelt, können Sie einen sogenannten Shebang (die Zeichenkombination **!#**) verwenden. Für Python sieht die Shebang-Zeile folgendermaßen aus, und Sie fügen ihn in die erste Zeile Ihrer Quelldatei ein: **#!/usr/bin/python**. Danach geben Sie **chmod +x hello.py** ein und können Ihre Datei ohne weitere Kommandos öffnen. Geben Sie einfach **./hello.py** ein, um das Skript aufzurufen.

Es ist nützlich, wenn Sie mit der Kommandozeile umgehen können, da diese einen guten Überblick bietet und einfach zu benutzen ist. Es gibt allerdings eine bequemere Möglichkeit der Befehlseingabe, nämlich die sogenannte Entwicklungsumgebung. Im Grunde handelt es sich dabei um nichts anderes als einen aufgemotzten Texteditor. Es gibt stets einen Bereich für die Eingabe von Text, aber häufig auch noch Bereiche für spezielle Features, die die Programmiersprache, in der Sie entwickeln, mitbringt. Zum Betriebssystem Raspbian gehört eine integrierte Entwicklungsumgebung (IDE) namens IDLE. Sie sollten ruhig ein bisschen damit herumprobieren, um herauszufinden, ob IDLE Ihre bevorzugte Entwicklungsumgebung werden könnte. IDLE startet im interaktiven Modus, deren große Vorteile darin bestehen, dass der eingegebene Code gemäß den unterschiedlichen Arten von Anweisungen spezifisch hervorgehoben wird, dass Pop-ups Informationen zu den bei Python verwendeten Objekten, Variablen und Methoden liefern und dass alle Einrückungen automatisch vorgenommen werden. Geben Sie beispielsweise zunächst **value = 1** in IDLE ein und danach **value.**, dann erscheint ein Menü, das Ihnen die möglichen Methoden zu der Variablen liefert.

Anweisungen

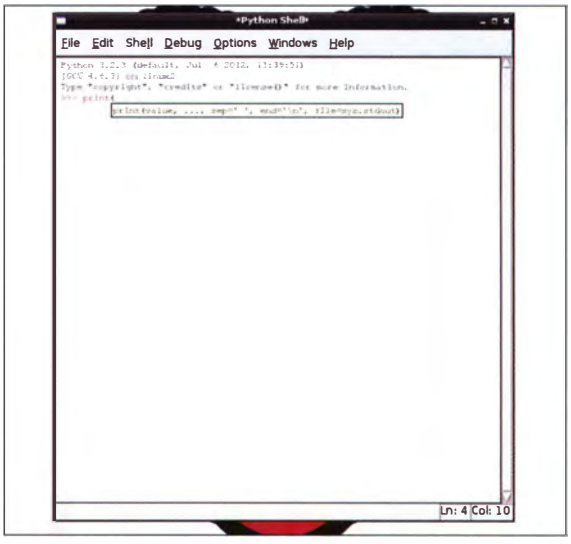
Nun da wir die Grundlagen von Python-Skripten kennengelernt haben, ist es an der Zeit, sich die Sprache selbst etwas genauer anzusehen. Wir haben bereits den Befehl print **("Hello world!")** ausgeführt, und dieser sagt uns schon viel über das Programmieren in Python. Vorn steht die Anweisung **print**. Anweisungen sind Wörter mit einer vordefinierten Bedeutung in Python, insgesamt gibt es nur einige wenige davon. Die Vorschrift **print** nimmt wie gesehen den in Klammern und Anführungszeichen stehenden Wert und übergibt ihn an das Standard-Ausgabegerät, was in den meisten Fällen der Bildschirm sein wird. Dies ist ein Beispiel für etwas, das Funktion genannt wird, und alles, was dabei geschieht, ist, dass Daten an einen anderen Teil des Codes übergeben und von diesem bearbeitet werden. **print** ist eine der wenigen in Python vorab definierten Anweisungen – gewöhnlich schreibt man eigene Anweisungen oder übernimmt welche von anderen Programmierern. Würden wir unsere „Hello world!"-Zeile in eine Funktion umwandeln, sähe diese folgendermaßen aus:

```
def helloworld():
        print("Hello world!")
        return
```

Geben Sie dies doch einmal direkt in den Interpreter ein, dann können Sie sofort mit der Ausgabe experimentieren. Bei dem oben angegebenen Text sehen Sie außerdem etwas, das Sie unbedingt beherzigen sollten, nämlich die Einrückung von Codezeilen. Die erste Zeile – in der die Funktion definiert wird – beginnt am linken Rand, und alle weiteren Zeilen der Funktion werden mit der Tab-Taste (oder mittels Leerzeichen) eingerückt. Das erleichtert die Lesbarkeit des Codes ungemein und erlaubt es dem Programmierer, auf einen Blick zu sehen, wo ein Block beginnt und endet. Wenn Sie nun **helloworld()** in eine neue Zeile tippen, sehen Sie die Ausgabe der soeben erstellten Funktion. Sie haben nun Ihren ersten logischen Block Code geschrieben!

Bedingte Anweisungen

Wenn Sie bisher noch nie programmiert haben, fragen Sie sich vielleicht, wie Sie die Logik in Ihrem Code aufbauen sollen. Dazu gibt es viele Möglichkeiten, doch die einfachste ist es, sich an die Reihenfolge zu halten, in der Bedingungen durchlaufen werden. Damit ist der Ablauf von Anweisungen zu einem Entscheidungspunkt, der von dem Zustand eines Parameters abhängt, gemeint. Der Ablauf kann in verschiedene Richtungen weitergehen. Der Grundstein für eine Bedingung ist **if**: »



› **Integrierte Entwicklungsumgebungen wie IDLE sind im Grunde Texteditoren mit erweitertem Funktionsumfang.**

```
if something == 1:
        print("It's true!")
```

Das ist das korrekte Format einer **if**-Anweisung in Python. Hier gibt es einiges Neues, etwa das Wort **something**. Es handelt sich hierbei um eine Variable, ihr Name ist beliebig gewählt. Eine Variable beinhaltet einen – eben *variierenden* – Wert, und im Fall der **if**-Anweisung wird geprüft, ob der Inhalt der Variablen der Wert 1 ist. Wenn dies der Fall ist, wird die Zeile „It's true!" ausgegeben. Ist dies jedoch nicht der Fall, wird die **print**-Anweisung übersprungen. Variablen sind extrem wichtig, und im Gegensatz zu anderen Programmiersprachen müssen Sie in Python nicht einem vordefinierten Datentyp zugewiesen werden. Um den oben genannten Codeschnipsel laufen zu lassen, müssen wir **something** einen Wert zuweisen:

```
something = 1
```

Vielleicht fragen Sie sich, warum wir diesmal nur ein einfaches Gleichheitszeichen in der Zuweisung benutzt haben und nicht – wie bei der **if**-Bedingung – ein doppeltes. Dies liegt daran, dass die beiden Varianten unterschiedliche Aufgaben haben, die man gut auseinanderhalten sollte. Das doppelte Gleichheitszeichen ist ein Vergleichsoperator. Es gibt daneben noch weitere Operatoren dieser Sorte:

<	kleiner
<=	kleiner gleich
>	größer
>=	größer gleich
==	gleich
!=	ungleich

Diese Operatoren können Sie benutzen, um den Wert einer Variablen zu überprüfen, während das einfache Gleichheitszeichen der Variablen einen Wert zuweist. Daher wird es Zuweisungsoperator genannt. Das = ist der häufigste Zuweisungsoperator, doch es gibt noch andere:

+=	Addition und Zuweisung
-=	Subtraktion und Zuweisung
*=	Multiplikation und Zuweisung
/=	Division und Zuweisung
%=	Zuweisung des Teilungsrestes

Diese Operatoren erlauben es Ihnen, Werte in Parametern zu manipulieren, wie Sie es aus mathematischen Formeln kennen. Und genau wie in mathematischen Formeln können Sie Klammern verwenden, um die Operationen in genau der Reihenfolge ablaufen zu lassen, die Sie wünschen. Zum Beispiel:

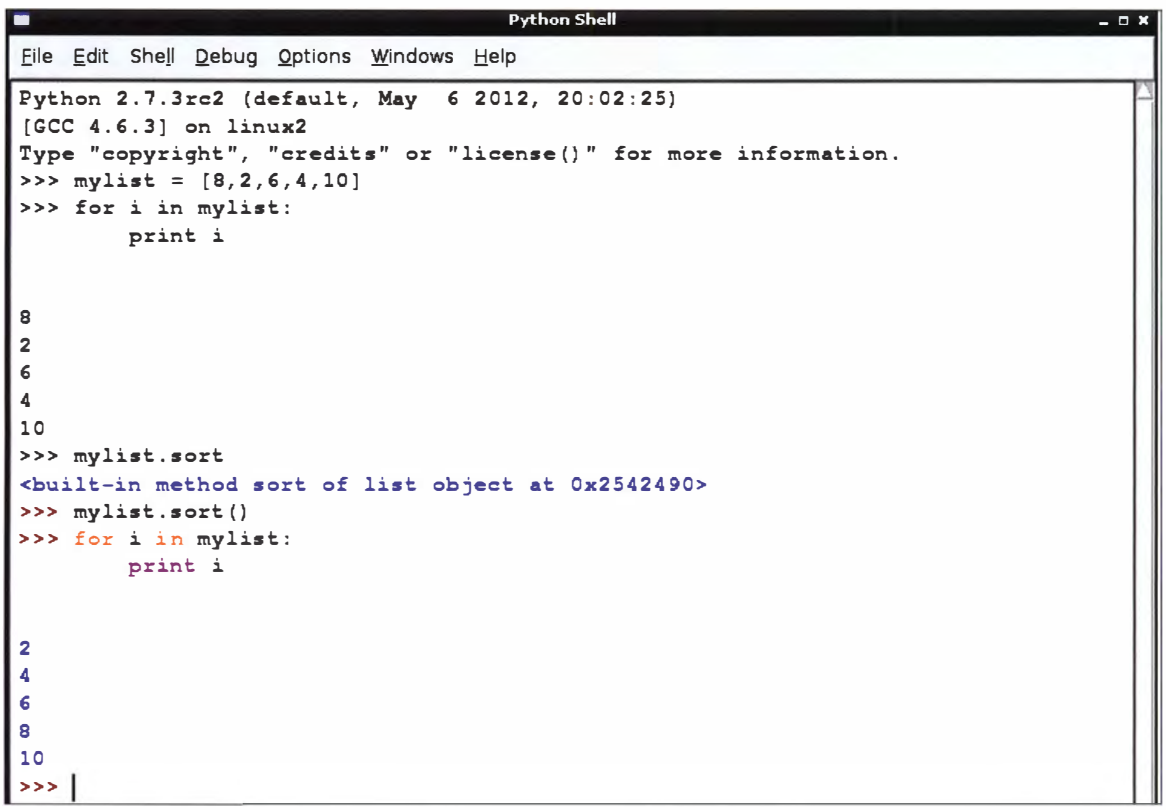
```
total = (10/2) * (20/10)
```

Geben Sie **print(total)** ein, um sich das Ergebnis anzeigen zu lassen und zu sehen, wie es berechnet wurde. Es gibt noch viele weitere Operatoren, aber für die meisten Projekte werden diese nicht notwendig sein. Haben Sie bemerkt, dass wir ein weiteres wichtiges Konzept in dem Beispiel behandelt haben? Wir übergaben **print** einen Variablennamen und keinen Wert, und zudem war der Wert der Variablen eine Zahl und kein alphanumerisches Zeichen. **print** war trotzdem in der Lage, daraus eine sinnvolle Ausgabe zu erstellen.

Die Fähigkeit, ein Argument in Klammern an eine Funktion zu übergeben, ist ebenfalls wichtig, und Sie können dies bei Ihren eigenen Funktionen genauso machen, indem Sie in der Funktionsdefinition einen Variablennamen in die Klammern schreiben. Um unsere „Hello world!"-Funktion vollständig allgemein zu halten, ändern wir sie folgendermaßen ab:

```
def helloworld(message):
        print(message)
        return
```

Wenn Sie nun **helloword("This is a test")** aufrufen, wird der Wert innerhalb der Klammern an die Variable „message" innerhalb der Funktion übergeben und per **print** ausgegeben. Dank der **print**-Funktion in Python brauchen wir uns im Gegensatz zu



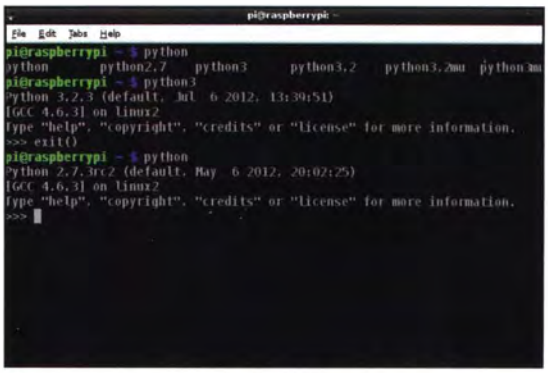
› Der for-Loop ist eine der am häufigsten vorkommenden Anweisungen. Ohne ihn kann man kaum ein Python-Skript schreiben.

Welche Python-Version habe ich?

Es gibt zwei Versionen von Python, die weit verbreitet sind – Version 2 und Version 3 (auch bekannt als Python 3000) –, was zu Verwirrung führen kann. Die am häufigsten verwendete Version ist die ältere Version Python 2.x, die lange Zeit der Standard für alle Python-Entwicklungsprojekte war. Das ist auch die Version, die der Raspberry Pi standardmäßig verwendet. Sie können das überprüfen, indem Sie **python** in die Kommandozeile eingeben. Die erste Ausgabezeile sollte im Prinzip ungefähr so aussehen:

```
Python 2.7.3rc2 (default, May 6 2012, 20:02:25)
```

Das bedeutet in diesem Fall, dass Sie Python 2.7.3 (release candidate 2) verwenden, was der letzte Stand des 2.x-Zweigs ist. Obgleich das Kommando **python** bei Raspbian zu dieser Version der Programmiersprache führt, ist Version 3 überraschenderweise ebenfalls standardmäßig installiert. Wenn Sie in die Kommandozeile lediglich **pyth** tippen und dann die Tab-Taste drücken, schlägt die Auto-Vervollständigung Ihnen alle weiteren möglichen Eingaben vor, etwa python2.7 und python3.2, die jeweils eine eigene Anwendungsdatei besitzen. Falls Sie Ihr Skript mit der neueren Python-Version starten möchten, geben Sie an dieser Stelle einfach **python3.2** ein. Analog dazu können Sie auf dem Raspbian-Desktop mit IDLE beziehungsweise IDLE3 auswählen, ob sie die Entwicklungsumgebung für Python 2.7 oder Phython 3 verwenden möchten. Zumindest am Anfang spielt es jedoch kaum eine Rolle, mit welcher Python-Version Sie arbeiten. Aus Kompatibilitätsgründen sind die meisten unserer Tutorials für die ältere Version geschrieben.



› Standardmäßig sind zwei Versionen von Python installiert. Per Kommandozeile können sie zwischen diesen auswählen.

vielen anderen Programmiersprachen keine Gedanken zu machen, welchen Datentyp eine Variable hat, bevor wir sie an **print** übergeben. Die einzelnen Datentypen sind trotzdem wichtig, weil sie vorgeben, wie Funktionen aufgebaut werden und was sie tun.

Datentypen

Auch wenn Sie einer Variablen keinen Datentyp zuweisen müssen, sind Datentypen bei Python genauso wichtig wie bei anderen Programmiersprachen. Das liegt daran, dass Datentypen normalerweise nicht von einem Typ in einen anderen Typ konvertiert werden können, ohne dass es zu Fehlern kommt. Wenn Sie eine lange Fließkommazahl wie Pi in eine Ganzzahl und dann zurück in eine Fließkommazahl konvertieren würden, gingen alle Nachkommastellen verloren. Das ist wichtig zu verinnerlichen, damit Sie schon im Vorfeld planen können, mit welchen Datentypen ihr Code arbeiten wird, bevor Sie den Code geschrieben haben. Dieses Wissen öffnet Ihnen auch den Zugang zu einigen Extrafunktionalitäten von Python.

Zum Beispiel haben wir in unserer „Hello world!"-Ausgabe bereits mit dem **string**-Datentyp gearbeitet, auch wenn wir ihn nicht explizit so benannt haben. Wenn Sie einer Variablen eine Zahl zuweisen, wird diese normalerweise eine Ganzzahl sein, solange sie kein Komma enthält, und andernfalls vermutlich eine Fließkommazahl. Abgesehen von den Datentypen, die einzelne Werte enthalten, gibt es spezielle Datentypen, die eine Auflistung von Werten enthalten können. Diese werden Arrays genannt. Arrays sind sehr nützlich, wenn Sie eine Gruppe von Werten zusammengruppieren möchten. Hier ist ein Beispiel:

```
mylist = [8,2,6,4,10]
```

Die obige Zeile erstellt eine Liste von Zahlen und weist sie der Variablen **mylist** zu. Wenn Sie bestimmte Elemente des Arrays ansprechen wollen, können Sie dies zum Beispiel über **mylist[0]** tun. So würden Sie das erste Element im Array ansprechen. Arrays beginnen immer mit einer 0, nicht mit einer 1.

Zum Schluss möchten wir noch zwei weitere Dinge veranschaulichen, bevor wir Sie in die Welt der Python-Tutorials entlassen. Das erste ist das Prinzip der Loops. Ein Loop führt den gleichen Abschnitt Code so lange aus, bis eine Bedingung erreicht wird. Wir können zum Beispiel jedes Element in unserer zuvor erstellten Liste mit einem solchen **for**-Loop ausgeben:

```
for i in mylist:
        print i
```

```
8
2
6
4
10
```

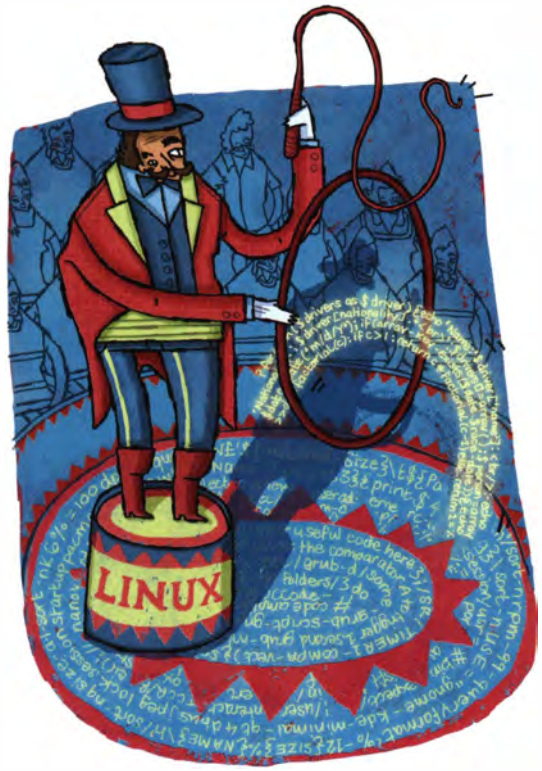
Alles, was der Loop tut, ist, für jedes Element in der Liste die nachfolgende **print**-Anweisung auszuführen. **for**-Loops werden sehr häufig verwendet, um Aufgaben dieser Art zu erledigen, und Sie werden sie bald als festen Bestandteil in Ihren Programmen verwenden. Ein genauso wichtiger Bestandteil sind Methoden. Es handelt sich dabei um Funktionen, die die von Ihnen verwendeten Datentypen erben. Bei unserer Beispielliste wäre **sort** eine solche Methode. Wenn Sie dem Beispiel vor der **print**-Anweisung den Methodenaufruf **mylist.sort()** hinzufügen, wird die Anordnung der Elemente im Array verändert. Lassen Sie den **for**-Loop jetzt laufen, so werden die Elemente in aufsteigender Reihenfolge ausgegeben.

> **„Es gibt bei Python jede Menge zu entdecken und auszuprobieren."**

Mit diesen Ausführungen haben wir Python nur ein klein wenig angerissen. Es gibt noch jede Menge mehr zu entdecken und auszuprobieren. Eine gute Informationsquelle ist die offizielle Website **http://docs.python.org**, und daneben gibt es noch viele andere Anlaufstellen im Internet, wenn Sie Unterstützung brauchen oder sich mit anderen Python-Anwendern austauschen möchten. ■

Online-Chat: Wir

Finden Sie, Ihr Leben könnte angenehmer sein? Dann folgen Sie dem Rat von Nick Veitch und programmieren Sie einen Chatbot, der Ihren Befehlen gehorcht.



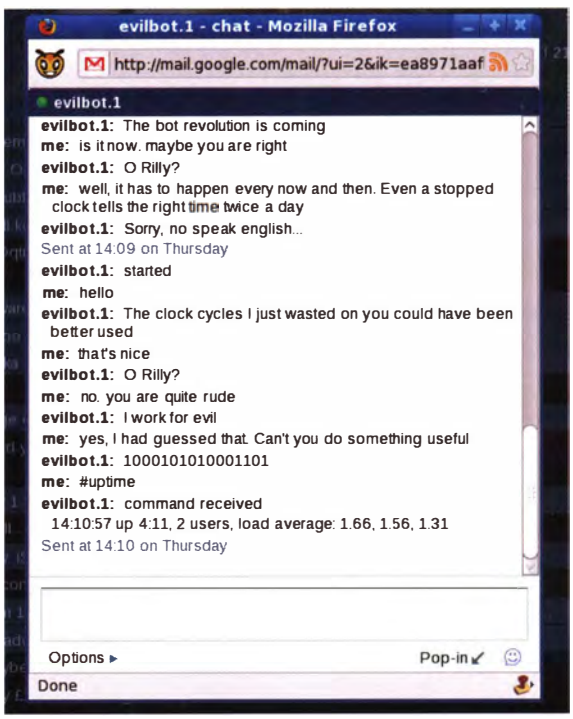
Stellen Sie sich vor, wie einfach das Leben wäre, wenn wir einige kleine Helfer hätten, die uns die Arbeit abnähmen: Die Post aus dem Briefkasten holen, die Wäsche waschen – all die lästigen Dinge eben, um die sich niemand reißt. Leider leben wir noch ein wenig zu früh in der menschlichen Geschichte, um bereits in den Genuss kostengünstiger robotischer Lakaien zu gelangen, sodass wir uns in vielen Fällen selber helfen müssen. Die nächstbeste Alternative wäre da ein virtueller Diener, der für uns allerlei Dinge erledigt. Natürlich erfüllen schon sehr viele Programme dieses Anforderungsprofil, doch ich denke im Speziellen an ein einfach zu bedienendes Interface, das nützliche Aufgaben erledigen kann und Ihnen nur Dinge mitteilt, die Sie auch wirklich wissen wollen. Wenn dieses Interface nun auch noch Ihre Freunde in Verwirrung stürzen und Ihre Feinde die Gründe für ihre Abneigung Ihnen gegenüber neu bewerten lassen würde – umso besser!

Einer der am wenigsten verwendeten – jedoch sicher bequemsten – Wege, mit so einem Diener zu kommunizieren, wäre über einen Chat. Warum sollte man sich auch mit komplizierten SSH-Tunneln oder mit langweiligen netzbasierten Programmen herumschlagen, wenn Sie auf einfache Art und Weise diese Aufgaben durch ein Medium erledigen könnten, das sie vermutlich ohnehin schon die ganze Zeit verwenden? Dies sei darum der Plan: einen kleinen Kobold in Form eines Chatbots zu erschaffen, der geduldig auf die Stimme seines Herrn oder seiner Meisterin in einem Chatkanal wartet, und sich von selber nur meldet, wenn es etwas Wichtiges zu berichten gibt.

Hallo, mein Name ist Xmpppy

Das Jabber/XMPP-Protokoll wird von Python gut unterstützt. Es existiert als Element des allumfassenden Netzwerkmoduls *Twisted*, jedoch gibt es auch eine etwas abgespeckte Version namens *Xmpppy*, die für unsere Zwecke vollkommen ausreichend ist. Sie sollten ohne Weiteres in der Lage sein, die notwendigen Pakete für Ihre Distribution zu finden, oder aber Sie laden sich einfach selbst den Python-Code von der Seite **http://xmpppy.sourceforge.net** herunter.

Um zu verstehen, wie Sie *Xmpppy* einsetzen können, beginnen wir einfach mit ein paar Beispielen für die Befehlszeile. Zuvor benötigen Sie jedoch einen Account. Sie könnten beispielsweise einen separaten Google-Account für Ihren Bot anlegen. Weiterhin brauchen Sie mindestens zwei Jabber-IDs für Testzwecke. In unserem Testlabor haben wir für diesen Zweck einen Gmail-Account für unseren Bot kreiert, uns via Browser eingeloggt und einen anderen Gmail-Account zu einem Chat eingeladen. Selbstverständlich können Sie all das auch innerhalb von *Xmpppy* abwickeln, aber für unsere ersten Gehversuche wäre dies viel zu kompliziert gewesen. Daher ist es vernünftiger,



› **Folgen Sie unseren Hinweisen, dann wird Ihr eigener Chatbot sehr schnell zu einer persönlichen Nervensäge.**

bauen Bots!

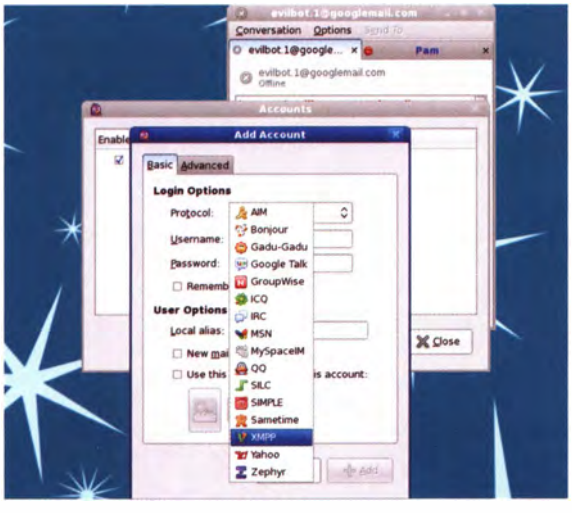
für die erste Phase zwei Accounts zu verwenden, die in der Lage sind, miteinander zu kommunizieren. Wenn Sie soweit sind, geben sie in der Konsole einfach **python** ein, um den interaktiven Kommandozeileninterpreter von Python zu starten.

```
>>> import xmpp
>>> jid=xmpp.protocol.JID("botadresse@googlemail.com")
```

Hier haben wir eine Instanz einer Jabber-ID eingerichtet. In diesem Fall handelt es sich um eine E-Mailadresse, da wir einen Google-Account in diesem Tutorial verwenden. Die Prozedur lässt sich aber mit jeder beliebigen Jabber-ID durchführen. Wir haben hier zwei Bereiche (es können auch drei sein, dazu später mehr), nämlich den Namen des Users sowie die Domain. *Xmpppy* versucht, in der Domain einen Server zu finden, den es ansprechen kann. Nachdem wir nun den User bestimmt haben, sollten wir ein Objekt erschaffen, das als Client fungiert. Der Client ist die Steuerungseinheit in *Xmpppy*, welche die Verbindungen kontrolliert, die Nachrichten verwaltet und generell mit dem Server interagiert. Wir müssen nur noch ein paar einfache Schritte vornehmen, um den Client einzurichten und eine Verbindung mit ihm aufzubauen. Wir müssen zunächst eine Instanz des Clients erschaffen (hierbei ist eine weitere Jabber-ID Voraussetzung), uns dann mit dem Server verbinden und uns zuletzt noch authentifizieren.

```
>>> myclient=xmpp.Client(jid.getDomain(),debug=[])
>>> myclient.connect()
'tls'
>>> myclient.auth(jid.getNode(),'botpasswort')
'sasl'
```

Hier gibt es einige Dinge zu beachten. Zuerst haben wir einen Client erschaffen und mittels des **jid.getDomain**-Befehls den Servernamen von unserem **jid**-Objekt, welches wir schon vorher angelegt hatten, in Erfahrung gebracht. Zusätzlich muss



› **Das XMPP/Jabber-Protkoll ist weit verbreitet, sodass Sie sich mit Ihrem Bot über zahlreiche Clients, inklusive Pidgin, verbinden können.**

Disco-Plugin

XMPP beinhaltet ein Set von Plug-ins mit dem Namen *Disco*. Hiermit werden die Nachrichtenprotokolle so erweitert, dass auch andere Typen von Nachrichten bearbeitet werden können, darunter SIP (Voice-Chat), Dateitransfer und alles mögliche andere, was über ein Punkt-zu-Punkt-Netzwerk übertragen werden kann. Es spricht nichts dagegen, Ihren Bot auch um diese Funktionen zu erweitern. Vielleicht könnte er Ihnen Speicherplatz zur Verfügung stellen oder aber wichtige Nachrichten vorlesen. Dankenswerterweise hat Google die Implementierung von XMPP noch um einige zusätzliche Funktionen erweitert, die Sie hier finden: **http://code.google.com/apis/talk/jep_extensions/extensions.html**

auch festgelegt werden, wie umfassend der Debugger Ihnen Rückmeldung gibt. Wir haben ein leeres Feld stehen lassen, aber falls Sie maximal informiert werden möchten, geben Sie stattdessen **always** ein.

Quick-Tipp

Sie erhöhen die Sicherheit Ihres Bots auf einfache Weise, indem Sie ihn nur Befehle von einer bestimmten ID annehmen lassen.

Verbindung einrichten

Nachdem wir das Debug-Level bestimmt haben, haben wir eine Verbindung (tragen sie dabei Ihr Passwort ein) aufgebaut und als Antwort die Meldung **tls** bekommen – dies bedeutet, dass eine sichere Verbindung hergestellt wurde. Die Ausgabe hätte auch **tcp** – für eine Standardverbindung – oder ein leerer String – für eine fehlgeschlagene Verbindung – sein können. Das Gelingen des nächsten Schritts hängt ein wenig von Ihrer Fingerfertigkeit auf der Tastatur und der Geschwindigkeit Ihres Servers ab, denn die Server von Google verlangen eine schnelle Authentifizierung. Um Zeit zu sparen, sollten Sie versuchen, die Kommandos in der Befehlszeile zusammenzufügen:

```
>>> myclient.connect() ; myclient.auth(jid.getNode
(),'botpasswort')
'tls'
'sasl'
```

Die Anwort **sasl** bedeutet, dass Ihre Verbindung via SASL (Simple Authentication and Security Layer) angenommen wurde. Der letzte Schritt, um Ihre Verbindung zu initialisieren, ist die Bekanntgabe Ihres Anwesenheitsstatus. Google-Chat und andere Jabber-Dienste sehen verschiedene Stati der Anwesenheit (anwesend, beschäftigt usw.) vor. Diese Funktion wird vor allem vom Server dafür verwendet, Listen über verfügbare Kontakte zu erstellen. Aber auch hierfür liefert Ihnen *Xmpppy* den richtigen Befehl:

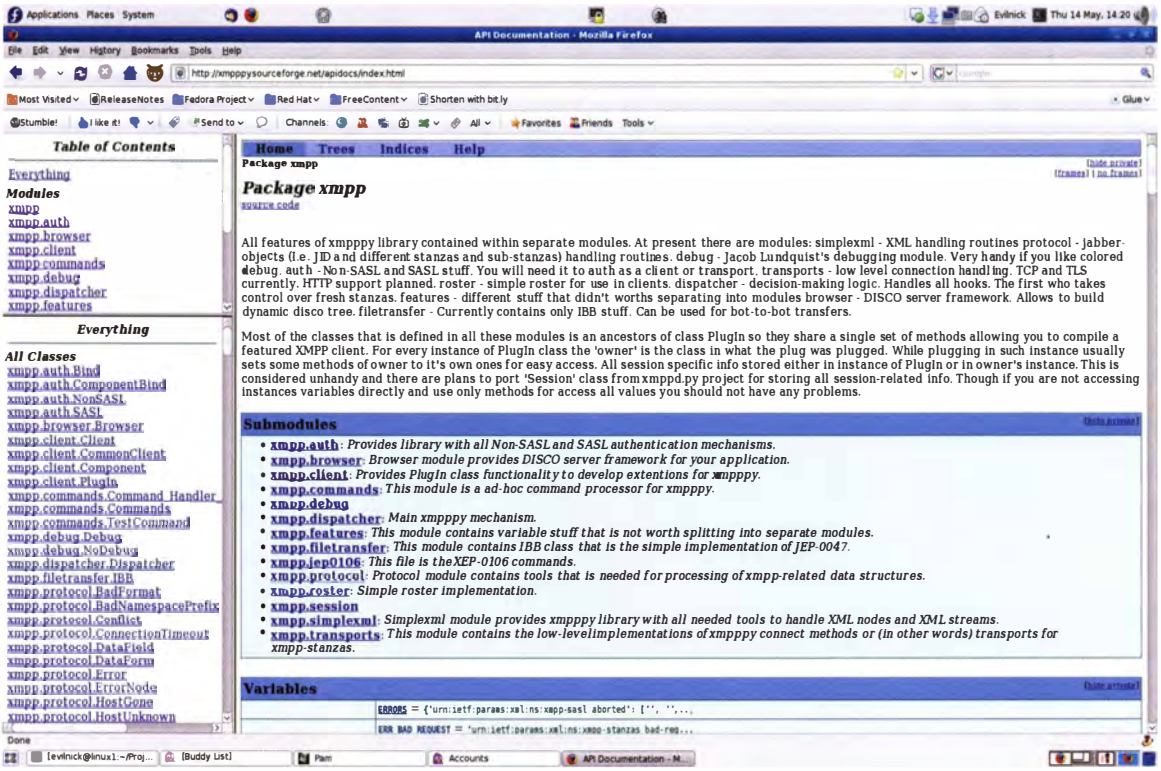
```
myclient.sendInitPresence()
```

Einige Server verwehren Ihnen den Zugriff, bis Sie einen Anwesenheitsstatus festgelegt haben!

Anwesenheitsstatus übermitteln

Endlich sind wir drin! Bevor es jetzt zu langweilig wird, schreiben wir eine Nachricht an unser Ziel (etwa Ihre eigene Google-Adresse). Damit es wirklich losgehen kann, brauchen wir natürlich noch die Empfängeradresse und den Inhalt unserer Nachricht. Haben wir das alles, können wir sogleich eine »

› **Die Projektseite von *Xmpppy* ist ein wenig schmal, was die Dokumentation angeht. Die API-Dokumente sollten aber voll und ganz ausreichen, um Ihnen zu zeigen, wie das Modul funktioniert.**



» Nachrichteninstanz erschaffen und diese abschicken:

```
mymsg=xmpp.protcol.Message("evilbot.1@gmail.com",
"hello","chat")
myclient.send(mymsg)
```

Vorausgesetzt, Sie haben es geschafft, die Chateinladung des Bots anzunehmen, sollten Sie nun eine kleine Überraschung erleben.

Falls Sie sich fragen, wie Sie auf dem Google-Server Ihren Anwesenheitsstatus festlegen können: Dies geschieht per XMPP, und zwar mithilfe einer speziellen Nachricht, die nur an den Server gesendet wird. Der Server verwaltet den Status und sendet die Information wiederum an jeden, der mit ihm verbunden ist.

```
>>> presence = xmpp.Presence(status = "Ready!", show =
"chat", priority = '1')
>>> myclient.send(presence)
```

Das *Xmpppy*-Modul verfährt etwas anders mit dem Verschicken der Anwesenheitsmeldungen, da die Parameter sich unterscheiden, aber prinzipiell ist der Mechanismus der gleiche.

Nachrichten empfangen

Um einen voll funktionsfähigen Bot zu erhalten, brauchen wir natürlich auch eingehende Nachrichten. Das bedeutet leider etwas Fummelarbeit. Das *Xmpppy*-Modul empfängt die Nachrichten über das von Ihnen erschaffene Client-Objekt und behält sie fürs Erste im Speicher, bis sie weiterverarbeitet werden. Die Verarbeitung geschieht bei *Xmpppy* mithilfe sogenannter Handler. Bevor Sie eine Nachricht verarbeiten können, müssen Sie erst einmal eine Funktion oder Methode definieren, die als Empfänger der Daten fungiert. Wenn sie bereit sind, die Nachrichten aus dem Speicher zu laden, aktivieren Sie einfach diesen Prozess auf dem Client-Objekt. Das klingt komplizierter, als es ist: Im Grunde heißt das nur, Sie sagen dem Client, wo er die Nachrichten hinschieben soll, und stupsen ihn dann an, damit er sie durch die erwähnte Funktion schickt. Man kann das Ganze per Python-Befehlszeile aufsetzen, allerdings wird es bei diesem Weg später ein bisschen unübersichtlich, da wir neben der Handler-Funktion noch eine Programmschleife benötigen. Bequemer ist es daher, für diesen Prozess ein eigenes Objekt zu erschaffen (dazu kommen wir noch). Hier zunächst die Befehlszeilen, mit denen wir eine Nachricht an uns selbst senden (Sie können alternativ auch Gmail benutzen).

```
>>> def msgparser(instance, message):
... print "Neue Nachricht!"
... print "from" + str(message.getFrom())
... print "msg" + str(message.getBody())
...
>>> myclient.RegisterHandler('message', msgparser)
>>> mymsg=xmpp.protocol.Message("evilbot@gmail.
com","hello","chat")
>>> myclient.send(mymsg)
'5'
>>> myclient.Process(1)
Neue Nachricht!
from: evilbot@gmail.com
msg: hello
1493
>>>
```

Die Parserfunktion ist noch relativ einfach gestrickt. Sie benutzt die zur Verfügung stehenden Methoden **getFrom()** und **getBody()** einer eingehenden Nachricht und verwandelt diese in

Hilfe zu Python

Wenn Python für Sie Neuland ist, Sie aber schon Erfahrung in anderen Programmiersprachen gesammelt haben, sollten Sie nicht vor allzu große Probleme stoßen, solange Sie sich daran halten, die Code-Zeilen ordentlich einzurücken. Auf der Python-Website finden Sie einen großen Fundus von Dokumentationen, die sich mit den Funktionen, der Syntax und den Modulen dieser Programmiersprache beschäftigen.

einen String, der auf der Konsole ausgegeben wird. Für einen echten Bot müssen wir natürlich den Absender in eine Variable packen, um so die Möglichkeit zu haben, für eine Antwort die Nachricht weiterzuparsen. Daher werden wir bei unserem Bot eine Syntax anwenden, welche Sonderbefehle, vor denen ein „**#**" steht, direkt an ihn weiterleitet. Das heißt, sollte eine Nachricht eingehen, die mit diesem Hash-Zeichen versehen ist, sollten wir sie irgendwie bearbeiten. Andernfalls antworten wir mit einer zufällig ausgewählten Nachricht aus einer Liste.

Eine Abkürzung nehmen

Einen Handler für Befehle hinzuzufügen, kann ziemlich beschwerlich und lästig werden. Um also Zeit und Mühe zu sparen, definieren wir eine Klasse, die eine Art Abkürzung enthält. Wir verwenden die **eval**-Funktion von Python, um eine Aufforderung an unsere Methode in der Klasse zu richten, die den gleichen Namen trägt wie der Befehl, der an sie gesendet wird. Hiermit schummeln wir ein wenig, jedoch sparen Sie dadurch auch ein bisschen Platz, und es erleichtert Ihnen das Hinzufügen weiterer Befehle – Sie müssen einfach nur eine neue Methode für diesen erschaffen. Um einen sauberer programmierten Bot zu kreieren, müssten Sie wahrscheinlich eher einen Handler-Mechanismus programmieren, damit zusätzliche Befehle in Instanzen der Klasse hinzugefügt werden können. Hier ist aber erst einmal die einfachere Variante:

```
def messageparse(self, instance, message):
m = message.getBody()
sender=message.getFrom()
if (m[0]=='#'):
self.log('command received')
# hier Sonderfälle einfügen
# allgemeiner Befehl - erwartet eine existierende Methode
und Benötigung einer Absender-ID
try:
eval('self.'+m[1:]+'(sender)')
except:
self.client.send((xmpp.protocol.Message(sender,'Das geht
leider nicht.')))
else:
# etwas sagen, um die Etikette zu wahren
self.client.send((xmpp.protocol.Message(sender,
random.choice(self.responses))))
```

Wie Sie sehen können, wird die eingehende Nachricht auf das Vorhandensein eines **#** hin überprüft, und falls das Zeichen gefunden wird, erschaffen wir einen Methodenaufruf vom Rest des Strings, überprüfen die ID des Absenders und versuchen, diese auszuführen. Die **try**- und **execute**-Struktur versucht hierbei, Ausnahmefälle wie zum Beispiel die Nichtexistenz der Methode abzufangen. Sollte kein passender Befehl gefunden werden, senden wir eine zufällige Nachricht aus der Liste **responses** zurück. Im wirklichen Leben müssen Sie hierfür natürlich das **random**-Modul importieren. Der Befehl **random.choice** nimmt wahllos einen Inhalt aus allem, was ihm zur Verfügung steht. Ein Befehls-Handler könnte folgendermaßen aussehen:

```
def uptime(self, sender):
import subprocess
p=subprocess.Popen(["uptime"], stdout=subprocess.PIPE)
r=p.communicate()[0]
self.client.send((xmpp.protocol.Message(sender,r,)))
```

Hier sollten wir einige Erklärungen nachliefern, da dieser Code Befehle bearbeitet, die auf der lokalen Maschine ausgeführt werden, auf der auch der Bot läuft. Am Anfang steht die Definition, welche die eigene Instanz akzeptiert (dies wird von Python vorausgesetzt), und die Information des Absenders, die der Nachrichten-Handler für uns herausgefiltert hat. Zusätzlich haben wir einen **subprocess** aus der Bibliothek von Python importiert, um einen Befehl lokal bearbeiten zu können. Auch benutzen wir etwas, was als Popen-Methode (siehe die Zeile, die mit **p=subprocess.Popen** beginnt) bezeichnet wird. Diese wird ausführlich auf **http://docs.python.org/library/subprocess.html** erklärt.

Kurz zusammengefasst leiten wir den auszuführenden Befehl weiter und starten eine Anfrage, dass der Standardausgang an einen Kanal angeschlossen werde. Danach sind wir in der Lage, diesen Ausgang über die **communicate**-Methode der Popen-Instanz abzufragen und die zurückgeschickte Antwort des Befehls zu bearbeiten. Die letzte Zeile verpackt dies alles und schickt die Informationen als Chat-Nachricht heraus. Somit können Sie ihren Server nun über einen Chat-Client steuern und sich von ihm auf die Nerven gehen lassen.

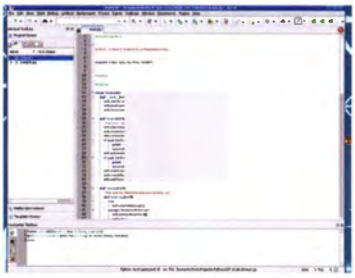
Ausblick

Womöglich möchten Sie noch eigene Methoden in die Klasse integrieren, um weitere Befehle zu definieren. Es gibt nichts, was Sie daran hindern sollte, diese Befehle dafür einzusetzen, Aufgaben zu automatisieren oder externe Quellen anzusteuern. Sie könnten zum Beispiel eine Erinnerungsfunktion für Ihren Online-Kalender implementieren. Oder verbinden Sie ihren Chatbot doch einfach mit einem Online-Übersetzungsdienst, sodass Sie ihn während einer Konversation als Dolmetscher verwenden können.

Der Bot ist eigentlich nicht viel mehr als ein Übertragungskanal – ein einfach zu bedienendes Interface, das es Ihnen ermöglicht, mit einem Script zu kommunizieren. Welche konkreten Aufgaben er letztendlich für Sie erledigt, bleibt Ihren Wünschen und Vorstellungen überlassen. ■

Python-Editoren

Python nimmt es ziemlich genau mit der Syntax. Das ist im Allgemeinen nicht schlimm. Es kann Sie aber vor Probleme stellen, wenn Sie versuchen, mit unzureichenden Editoren zu programmieren. Wir benutzen *Vim* und *Kate*. Beide haben – neben einigen anderen nützlichen Features – die Funktion, die Syntax hervorzuheben, sodass es mit ihnen einfach von der Hand geht, Scripts in Python zu programmieren. Sie können auch *Eric*, eine auf Python basierende IDE verwenden, die mit vielen pythonspezifischen Funktionen aufwartet. Das Download-Paket für die gebräuchlichsten Distributionen finden sie hier: **http://eric-ide.python-projects.org/eric4-download.html**



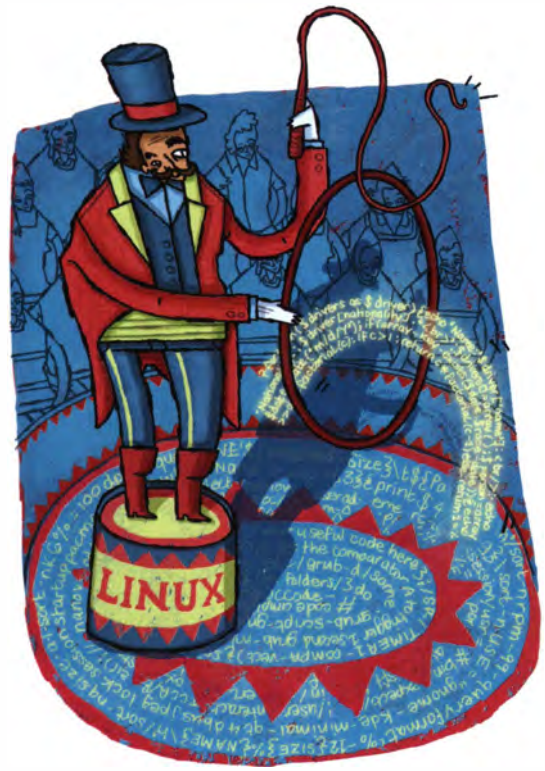
› **Benutzen Sie entweder einen Editor mit einer Hervorhebungsfunktion für die Python-Syntax, oder holen Sie sich die speziell für diese Sprache konzipierte IDE namens Eric.**

Versionen von Python

Mittlerweile gibt es Python 3.0, aber da keine Abwärtskompatibilität mit früheren Versionen besteht, benutzen viele Distributionen bis auf Weiteres die Version 2.x als Standard. Der Code in unseren Tutorials ist 2.x-kompatibel, damit er für die meisten User benutzbar ist.

Twitter: Tweets

Nick Veitch erschafft ein Monster – halb Python, halb Twitter. Und es spricht!



Twitter ist eine großartige Informationsquelle, wird von vielen jedoch auch als gigantische Ablenkungsmaschine betrachtet (was einander nicht ausschließt). Man kann damit allerdings auch eine Menge technische Dinge anstellen. Dazu muss man sich jedoch zuerst ein wenig mit der API (Programmierschnittstelle) von Twitter vertraut machen. Diese ist ein ganz schönes Sammelsurium – es scheint so, als hätte sich die API entwickelt, indem im Laufe der Zeit viele unterschiedliche Lösungswege für ähnliche Aufgabenstellungen gegangen worden wären. Das soll uns aber nicht von der Arbeit abhalten, denn es gibt zahlreiche API-Wrapper (Programmadapter) für Python. Für unsere Zwecke ist der Wrapper *Python-Twitter* gut geeignet, der auf **http://code.google.com/p/python-twitter** zu bekommen ist.

Alternative zu Twitter

Bevor wir weitermachen, eine kurze Frage: Haben Sie schon mal von identi.ca gehört? Das ist ein Mikro-Blogging-Dienst, ähnlich wie Twitter, nur dass dieser auf freier Software basiert und die Copyleft-Lizenz GPL gewährt. Unser Experiment funktioniert auch mit identi.ca. Zwar gibt es kein Python-Modul für den Dienst, aber identi.ca benutzt fast die gleiche API wie Twitter, darum kann man eigene Programme leicht anpassen. Man braucht nur die Serververbindungen in der Datei **twitter.py** entsprechend auf **identi.ca** zu ändern. Doch das ist eine andere Geschichte und soll ein andermal erzählt werden.

Mit Twitter verbinden

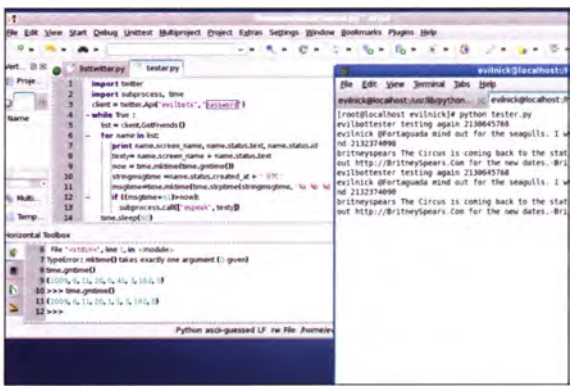
Um mit Twitter arbeiten zu können, benötigen Sie natürlich ein Benutzerkonto bei dem Dienst. Falls Sie bereits eines besitzen, können Sie sich für dieses Experiment auch ein weiteres anlegen, wenn Sie möchten. In dem Fall sollten Sie jedoch zunächst einige Kontakte knüpfen, um die Twitter-Mechanismen auf der Programmierebene richtig testen zu können. Für dieses Tutorial haben wir das Benutzerkonto „evilbotx" angelegt und sind damit ein Follower des Piratenschrecks Britney Spears geworden. Öffnen Sie ein Terminal, starten Sie Python mit dem Kommando **python**, und geben Sie danach Folgendes ein:

```
>>> import twitter
>>> client = twitter.Api(username="evilbotx",
 password="mypassword")
>>> client.PostUpdate("Hello World!") <twitter.Status object
at 0xb7c2f44c>
```

Dies ist unser normaler Weg der Benutzeranmeldung. Zunächst wird ein Objekt namens **client** geschaffen, welches sich dann mit dem Twitter-Server verbindet und authentifiziert, und zuletzt wird mithilfe einer Objektmethode ein Statusupdate vorgenommen. Schon geht es also los mit dem pythongesteuerten Mikro-Blogging!

Falls Sie ein komplett autonomes System bauen wollten, könnten Sie es hierbei bewenden lassen. Sie könnten diese Funktionalität in ein anderes Skript integrieren und jederzeit Tweets absetzen. Allerdings möchten wir gerne über diese Grundfunktion hinausgehen. Der nächste Schritt wäre dann, eine Liste der Twitterer anzulegen, denen wir folgen möchten. Dies ist nicht besonders kompliziert, denn für die meisten Funktionen stehen Methoden zur Verfügung:

```
>>> userlist = client.GetFriends()
>>> for username in userlist: ... print username.screen_name,
```



› **Wenn ein Bild mehr sagt als tausend Worte, warum höre ich dann nichts? Keine Sorge, das hier *kann* sprechen!**

vorlesen lassen

```
username.status.text ...
evilnick @tweeny4 it's hard to beat a poached egg
serenajwilliams @celebsdontreply. Of course, I reply.
britneyspears The Circus is coming back to the states -Britney
```

An diesem Stück Code können wir sehen, dass die **GetFriends()**-Methode eine Liste der Klasse **user** zurückbringt. Bei **user** handelt es sich um eine im Twitter-Modul festgelegte Klasse, an die mehrere Dinge wie etwa der Twitter-Name, die Kurzbiographie oder eine URL angehängt werden. All diese Informationen werden direkt von Twitter geliefert, wenn die Objekte erzeugt werden. Einige der Eigenschaften sind:

- **user.id**: Eindeutige Nutzer-ID
- **user.name**: Echter Name des Benutzers*
- **user.screen_name**: Twitter-Name des Benutzers
- **user.description**: Kurzbiographie des Benutzers*
- **user.profile_image_url**: Link zum Profilbild
- **user.url**: URL-Angabe des Benutzers, etwa zu dessen Website*
- **user.status**: Neuester Status des Benutzers

Die mit dem * markierten Eigenschaften können auch leer sein, wenn der Benutzer hier Twitter gegenüber keine Angaben gemacht hat.

Sprachausgabe von Tweets

Wir könnten diese Eigenschaften in unserem Code verwenden, beispielsweise, um Bilder für einen grafischen Twitter-Client zu holen oder Twitter-Benutzer nach Interessen zu ordnen.

Aber wie cool wäre es erst, einen audiobasierten Twitter-Client zu haben! Statt die Augen, die man gerade auf seinen Programmcode gerichtet haben sollte, mit dem Lesen von Twitter-Meldungen abzulenken, würde uns eine automatische Stimme die Kurznachrichten vorlesen! Es gibt einige Utilities, die geschriebene Texte in gesprochene Sprache umwandeln, und es gibt sogar ein Sprach-Dispatcher-System für Linux. Vielleicht gehören *Festival* oder *Espeak* bereits zum Lieferumfang Ihrer Distribution, ansonsten können Sie eins der Programme über die Repositorys nachinstallieren. In unserem Beispiel benutzen wir *Espeak*, aber der entsprechende Code wäre bei den anderen Programmen sehr ähnlich.

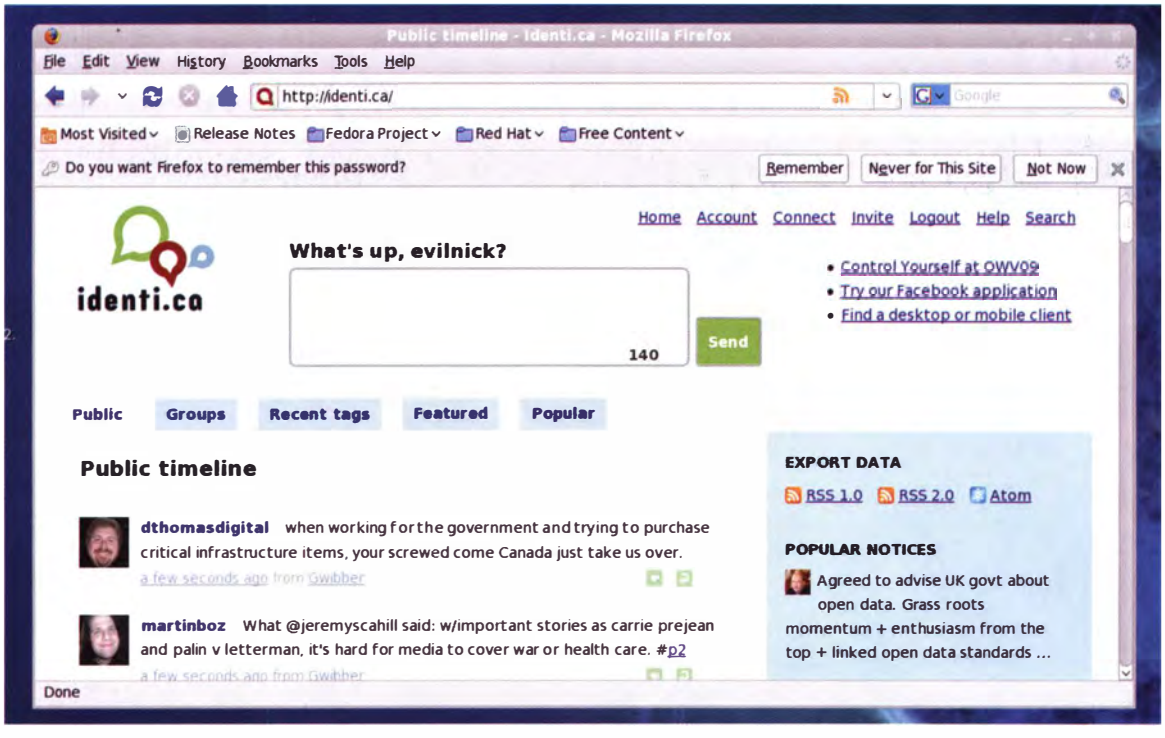
Wir brauchen uns bei dieser relativ einfachen Aufgabenstellung auch nicht mit komplizierten Modulen zu befassen, stattdessen nehmen wir das bewährte **subprocess**-Modul. Dieses »

identi.ca

identi.ca ist ein auf freier und Open-Source-Software aufbauender Mikro-Blogging-Dienst so wie Twitter. Die Inhalte stehen unter der Creative-Commons-Lizenz. Allerdings ist identi.ca in puncto Popularität meilenweit von Twitter entfernt, wobei das ja auch wieder ein Vorteil sein kann. Prinzipiell lässt sich die in diesem Tutorial gezeigte Vorgehensweise (mit einigen Anpassungen) auch auf identi.ca übertragen, denn dessen API ist sehr eng an die API von Twitter angelehnt.

Quick-Tipp

Sie können Sich jederzeit Informationen zur Funktionalität eines Ihnen nicht vertrauten Moduls anzeigen lassen, wenn Sie im Terminal **help (Modulname)** eingeben, nachdem Sie das Modul importiert haben.



identi.ca hat eine ähnliche Funktionalität wie Twitter, ist aber längst nicht so verbreitet. Probieren Sie die Plattform doch einmal aus.

» ermöglicht es, innerhalb von Python Kommandozeilenprogramme aufzurufen. Wir benutzen die call-Methode, die lediglich eine Liste der Parameter benötigt, die Sie verwenden möchten. Ein einfaches Beispiel ist:

```
>>> import subprocess
>>> subprocess.call(['espeak','"Hello World!"'])
```

Daraufhin sollte Sie eine Computerstimme begrüßen. Falls Sie einen Syntaxfehler angezeigt bekommen, überprüfen Sie, ob Sie möglicherweise bei den Anführungszeichen etwas Falsches getippt haben. Das letzte Element in der Liste ist ein Textstring innerhalb von doppelten Anführungszeichen, der wiederum als Ganzes innerhalb von einfachen Anführungszeichen steht. Der Code entspricht dem Eintippen des Befehls **espeak "Hello World!"** im Terminal. Ein funktionierender Client würde etwa so aussehen:

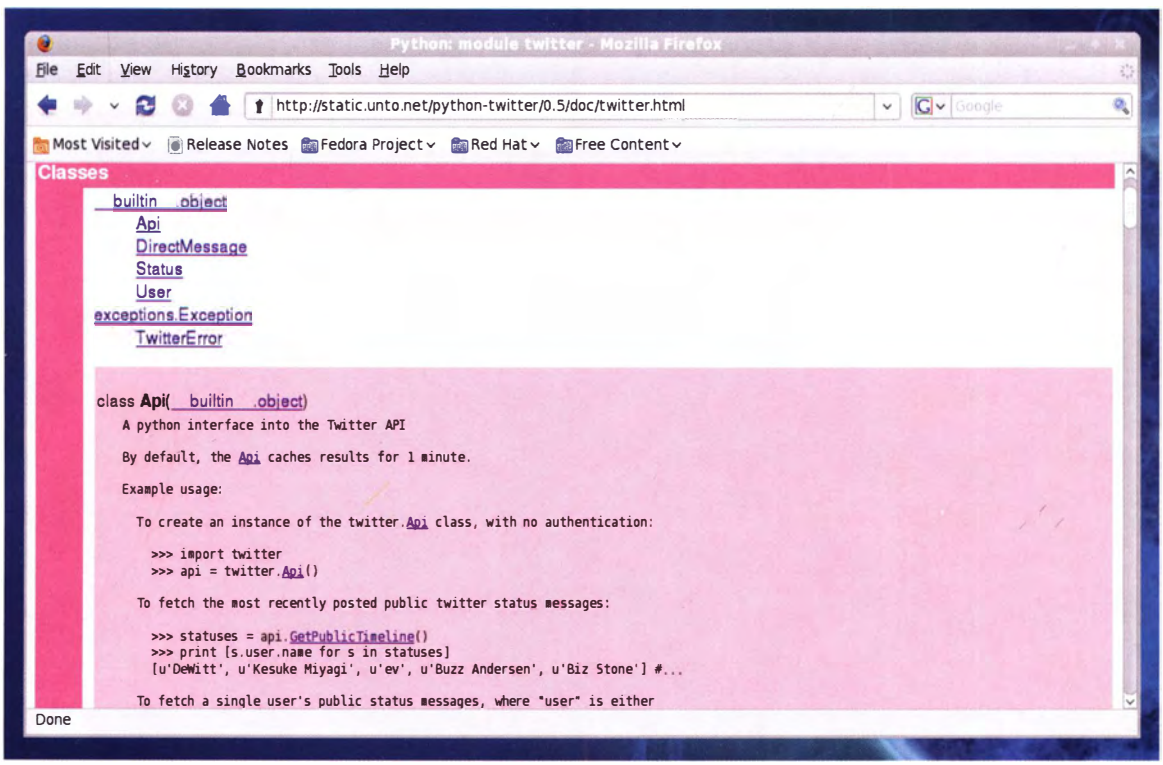
```
import twitter, subprocess, time
client = twitter.Api("evilbotx", "evilbot")
while True :
    list = client.GetFriends()
    for name in list:
        print name.screen_name, name.status.text, name.
status.id
        texty= name.screen_name + name.status.text
        time.sleep(2)
        subprocess.call(["espeak", texty])
    time.sleep(60)
```

In den obenstehenden Programmzeilen stellen wir eine Verbindung mit Twitter her, beginnen eine Endlosschleife und bekommen eine Liste von Kontakten ausgegeben. Eine weitere Schleife verarbeitet die Statusmeldungen und gibt sie aus, wandelt die Informationen, die wir haben möchten, in einen String (eine Zeichenkette) um und benutzt dann **subprocess.call**, um sie an die Sprachausgabe zu schicken. Am Ende des Skripts haben wir eine Verzögerung **time.sleep(60)** eingebaut, damit der Server nicht zu häufig abgefragt wird.

Vielleicht fragen Sie sich, warum wir die Abfrage der Kontaktliste aus dem Hauptloop heraus durchgeführt haben. Dies haben wir so programmiert, weil es die Sache in zweifacher Hinsicht simpler macht: Erstens werden allen Objekte der Klasse **user** automatisch die neuesten Statusmeldungen zugeordnet. Wenn wir die Kontaktliste nur einmal laden würden, müssten wir sie trotzdem bei jedem Durchlauf der Schleife durchgehen, um die Statusmeldungen zu erhalten. Und zweitens erlaubt uns diese Variante, gefahrlos einen zweiten Twitter-Client laufen zu lassen. Jede Änderung, die Sie an Ihrer Kontaktliste vornehmen, wird automatisch von diesem Skript erfasst.

Zeitprobleme

Nun haben wir einen funktionierenden Programmcode, doch es gibt immer noch ein Problem. Die Statusmeldungen werden hierbei in jedem Fall vorgelesen, ob sie sich nun geändert haben oder nicht. Wir könnten jetzt den Zeitpunkt der Veröffentlichung einer Statusmeldung mit der aktuellen Zeit vergleichen, und nur, falls eine Meldung vor weniger als 60 Sekunden gepostet wurde (oder sagen wir 61, um die Zeit des Programmdurchlaufs einzukalkulieren), dann soll sie vorgelesen werden. Leider ist Zeit jedoch relativ. Das Zeitmodul von Python kann Ihnen die derzeitige Epochenzeit angeben (dabei handelt es sich um die Anzahl der vergangenen Sekunden seit dem Anbeginn der Zeit – wobei dies aus Sicht von Unix der 1. Januar 1970, null Uhr ist), aber Twitter gibt die Veröffentlichungszeit eines Tweets in einem Textformat an. Um beides vergleichen zu können, müssen wir einige Maßnahmen ergreifen. Anscheinend trifft *Python-Twitter* eine fehlerhafte Annahme bei der Übersetzung, denn bei den Methoden zur Bestimmung des Zeitpunkts des Abrufs und der Veröffentlichung eines Tweets scheint etwas nicht übereinzustimmen. Das verkompliziert die Sache, ist jedoch nicht unlösbar. Die Twitter-API gibt Datum und Uhrzeit als



› **Eine vollständige Erläuterung des Moduls *Python-Twitter* finden Sie unter http://static.unto.net/python-twitter/0.5/doc/twitter.html**

Textstrings im Format **Mon Nov 25 11:46:34 +0000 2013** aus. Das ist an sich kein Problem, da Python dieses Format in sein übliches numerisches Format übersetzen kann und im nächsten Schritt in die Anzahl der Sekunden seit der Unix-Epoche umwandelt. Allerdings gibt das Twitter-Format keine konkrete Zeitzone an. Wenn wir diese mit der koordinierten Weltzeit UTC festlegen möchten, können wir einfach ein **UTC** an das Ende des Strings setzen und Python das Ganze in ein bequemeres numerisches Format parsen lassen. Das Objekt **status** bewahrt die Twitter-Zeitangabe in der Eigenschaft **created-at** auf, also können wir damit das Problem bei der internen Umrechnung umgehen. Die Funktion **time.strptime** verarbeitet Strings zu numerischen Werten in einer Standardform. Um dies zu erreichen, müssen wir der Funktion den String mitteilen sowie einen String, der das Format vorgibt. Der zweite String enthält Anweisungen oder Beschreiber gemäß der Liste von Werten, die das Modul akzeptiert. In unserem Fall sind dies **%a**: der abgekürzte Name des Wochentags, **%b**: der abgekürzte Name des Monats, **%d**: der Tag des Monats, **%H**: die numerischen Stunden, **%M**: die numerischen Minuten, **%S**: die numerischen Sekunden, **%Y**: das numerische Jahr und **%Z**, ein String aus drei Zeichen, der die Zeitzone repräsentiert.

Wie Sie sehen, haben wir den letzten Wert selbst hinzugefügt, sodass er aufgenommen werden kann, wenn Zeitangaben innerhalb von Python bearbeitet werden. Das von Python intern verwendete numerische Format drückt alle diese Daten in Zahlen aus, wie im nachfolgenden Output zu sehen ist:

```
>>> time.strptime('Mon Jun 8 10:51:32 +0000 2009 UTC', '%a
%b %d %H:%M:%S +0000 %Y %Z')
(2009, 6, 8, 10, 51, 32, 0, 159, 0)
```

Diese Zahlen stehen für das Jahr, den Monat, den Tag des Monats, Stunden, Minuten, Sekunden, den Wochentag (0 steht für Montag), den Tag des Jahres sowie Sommerzeit oder nicht. Es ist deshalb sehr wichtig, die Zeitzone hinzuzufügen, weil Python versucht, die Sommerzeit selbsttätig zu ermitteln, falls diese Angabe fehlt, was zu falschen Ergebnissen führen kann.

Diese Zeitdarstellung kann danach mit **time.mktime()** wieder in das übliche Unix-Sekundenformat seit der Epoche konvertiert werden. Umfassende Informationen zum **time**-Modul finden Sie auf **http://docs.python.org/library/time.html**.

Unser überarbeiteter Code sieht nunmehr wie folgt aus:

```
import twitter, subprocess, time
client = twitter.Api("evilbotx", "password")
while True :
    list = client.GetFriends()
    for name in list:
        texty= name.screen_name + name.status.text
        now = time.mktime(time.gmtime())
        stringmsgtime =name.status.created_at + ' UTC'
        msgtime=time.mktime(time.strptime(stringmsgtime,
'%a %b %d %H:%M:%S +0000 %Y %Z'))
        if ((msgtime+61)>now):
        subprocess.call(["espeak", texty])
    time.sleep(60)
```

Jetzt haben Sie also in einigen wenigen Codezeilen einen funktionierenden audiobasierten Twitter-Client geschrieben! Ein Problem bleibt allerdings noch: Falls Sie einer größeren Anzahl von Twitterern folgen möchte, könnte es schnell anstrengend werden, wenn ständig neue Postings vorgelesen werden. Möglicherweise wollen Sie die Liste derjenigen Kontakte, bei denen Meldungen vorgelesen werden sollen, ja verkleinern. Dann müssten Sie noch ein paar kleine Änderungen vornehmen:

Chat und Twitter

Wenn Sie das Tutorial auf Seite 94-97 durchgearbeitet und sich ein paar Chatbots zugelegt haben, dann könnten Sie die entsprechenden Skripte mit Ihren Twitter-Experimenten verknüpfen. Zum Beispiel, indem Sie Ihren Chatstatus aus Ihrem letzten Twitter-Posting generieren.

```
import
xmpp,twittertwituser="foo1"twitpass="foo2"jabberuser="bar1@something"jabber
pass="bar2"twit=twitter.Api(username=twituser,password=twitpass)text=twit.
GetUser(twituser).status.textjid=xmpp.protocol.JID(jabberuser)jab=xmpp.
Client(jid.getDomain(),debug=[])
jab.connect()
jab.auth(jid.getNode(),jabberpass)
jab.sendInitPresence()jab.send(xmpp.Presence(status = text , show = "chat" ,
priority = '1')
///end//
```

Platzieren Sie den gesamten Code in einer Schleife mit einer angemessenen Verzögerungszeit. Die Nutzernamen und Passwörter müssen Sie natürlich noch individuell anpassen.

```
import twitter, subprocess, time
client = twitter.Api("evilbotx", "password")
list = ['evilnick', 'evilbottester', 'tweeny4']
while True :
    for item in list:
        name=client.GetUser(item)
        texty= name.screen_name + name.status.text
        now = time.mktime(time.gmtime())
        stringmsgtime =name.status.created_at + 'UTC'
        msgtime=time.mktime(time.strptime(stringmsgtime,
'%a %b %d %H:%M:%S +0000 %Y %Z'))
        if ((msgtime+61)>now):
        subprocess.call(["espeak", texty])
    time.sleep(60)
```

In dieser Version des Skripts geht die innere Schleife die Liste durch und ruft die **GetUser()**-Methode für jeden Twitter-Namen auf. Dies führt zu einer Rückmeldung über **user**-Objekte mit bestimmten Eigenschaften, zu denen die neueste Statusmeldung gehört. Nun bekommen Sie nur noch die neuen Tweets Ihrer Favoriten vorgelesen.

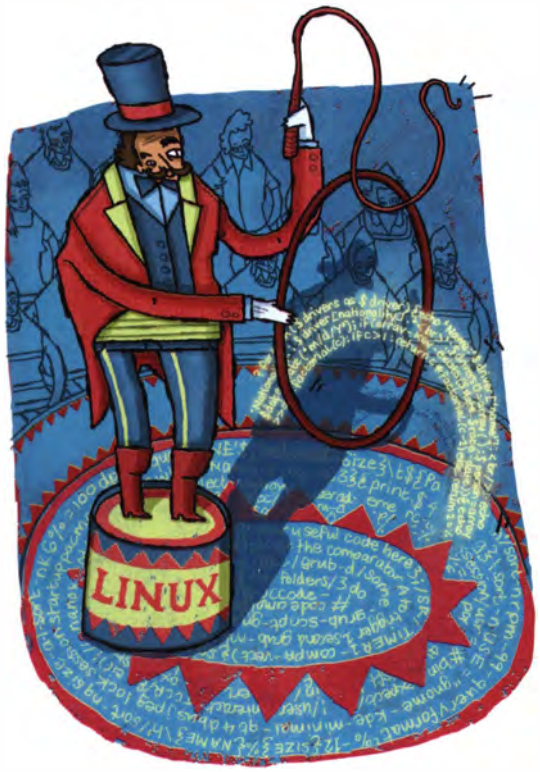
Ausblick

Es könnte eine nützliche Sache sein, eine grafische Benutzeroberfläche zur Hand zu haben, wenn Sie schnell Ihren eigenen Status aktualisieren möchten. Wenn Sie dazu *PyQt*, *wxWidgets* oder eine andere GUI verwenden, müssten Sie dazu lediglich einen Text-Input mit maximal 140 Zeichen erstellen und eine Methode zum Posten von Statusmeldungen an die Betätigung der Eingabetaste koppeln.

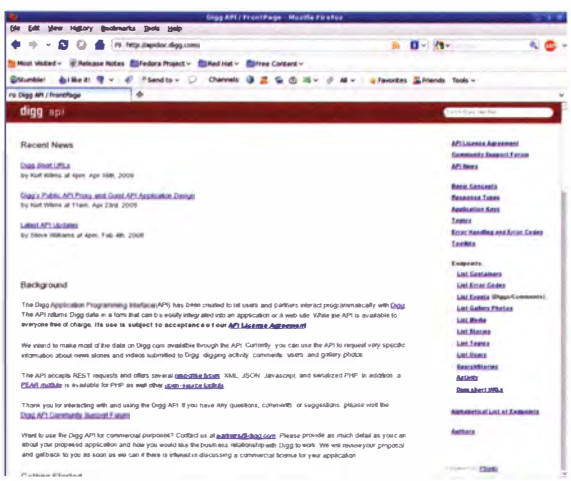
Eine andere Idee wäre es, Ihren Server seinen freien Speicherplatz twittern zu lassen oder das Skript der Software *Amarok* anzufügen und es twittern zu lassen, welche Musik Sie gerade hören. ■

Digg: API-Modul

In dieser Anleitung zeigen wir Ihnen, wie Sie eine API für Digg kreieren und Funktionalitäten hinzufügen.



Bislang haben wir in diesem Abschnitt des Handbuchs gezeigt, wie man mit bestehendem API-Code Web-Objekte nach eigenen Vorgaben verändern kann. Manchmal benötigt man auch nur einen kleinen Teil einer bestehenden API. Daher wollen wir nun eine eigene API programmieren, obwohl es ja bereits eine Schnittstelle für Digg gibt. Natürlich können wir nur anhand eines Beispiels die Möglichkeiten aufzeigen – eine vollständige API mit allen denkbaren Funktionen können Sie aber natürlich im Laufe der Zeit selbst erstellen. Viele Webseiten nutzen ein einziges Kommunikationsprotokoll für eine API, andere Webseiten nutzen gar mehrere. Die derzeit populärsten sind JSON und XML. JSON ist im Vergleich zu XML etwas schlichter und lässt sich leichter in JavaScript implementieren. XML ist etwas „sperriger" und größer, dafür aber auch eleganter. Man kann sich auch etwas einfacher in XML hineindenken, allerdings erhalten Sie für eine simple Abfrage eine Vielzahl von Antworten. Trotz des Umfangs werden wir im weiteren Verlauf auf XML setzen, ein elementarer Unterschied zwischen beiden Varianten besteht ohnehin nicht. Da XML auch deutlich weiter verbreitet ist, sollten Sie, wenn Sie mit API-Aufrufen via XML umgehen können, auf jeden Fall XML verwenden. Python kommt gut mit XML zurecht und besitzt ein etabliertes Modul zur Bearbeitung von XML. Im ersten Schritt müssen wir allerdings herausfinden, wie man mit der Digg-API interagieren kann. Zum Glück gibt es für die Digg-API umfassende Dokumentationen und Erklärungen.



› **Die Digg-API besitzt eine eigene Website, allerdings ist diese für Anfänger etwas unübersichtlich.**

API via URL-Abfrage

Besuchen Sie nun die Digg-API-Webseite unter **http://apidoc.digg.com** und schauen Sie sich etwas um. Die API wird über die Hauptseite durch Anfragen aufgerufen. Vermutlich sind Sie solchen Anfragen bereits begegnet, sie beginnen mit einem „**?**" und werden voneinander mit einem „**&**" getrennt. Sie müssen für die ersten Tests noch nicht einmal eigene Codezeilen schreiben, denn es genügt eine angepasste URL für eine Abfrage über den Browser. Da wir die Standard-XML-Antworten verwenden werden, genügt ein Browser wie Firefox, der den resultierenden XML-Code im Browserfenster anzeigen wird. Dies ist ein echter Vorteil, da Sie nicht erst den Umweg über eine Datei nehmen müssen. Auch der Umweg über einen Texteditor entfällt auf diese Art und Weise. Die Digg-API stützt sich auf definierte Endpunkte oder URL-Pfade, um bestimmte Funktionen zu aktivieren. Wenn Sie also die aktuelle Hot List finden wollen, geben Sie als URL **http://services.digg.com/stories/hot** ein.

Mit diesem Format erhalten Sie allerdings eine Fehlermeldung. Digg verlangt nämlich einen sogenannten „Appkey" oder „API key", um die gewünschte Funktion bereitzustellen. Diese Praxis ist bei vielen Webdiensten eine Grundvoraussetzung, da man so einen bestimmten Client ohne großen Aufwand identifizieren kann. Dies ist keine Gängelung, sondern dient dem Schutz der API. So will man beispielsweise Abfragen im 10-Millisekunden-Abstand unterbinden können, ohne die ganze API abklemmen zu müssen. Digg setzt diese Schutzmaßnahmen allerdings etwas offener um, denn Sie müssen keinen API-Key im Voraus registrieren, sondern Sie müssen eine URL angeben, die auf die Quelle der Client-Software hinweist.

Versuchen Sie die Anfrage hingegen in der Form **http://services.digg.com/stories/hot/?appkey=http://linuxformat.co. uk**, sollten Sie eine Ausgabe mit strukturiertem

selbstgemacht

Text zurückerhalten. Das „**?**“ in der URL markiert eine Anfrage, der wir einige Werte hinzufügen. Das gewöhnliche Format für dies ist eine Liste von **key=value**-Paaren, die von „**&**“-Zeichen getrennt werden. Wenn wir nun die URL nach **http:// services.digg.com/stories/hot/?appkey=http://linuxformat.co.uk&count=1** abändern, wird bei diesem Durchgang nur die erste Story angezeigt. Natürlich sind auch andere Zahlen und Variablen bei dieser definierten Abfrage möglich. Sie sind allerdings darauf angewiesen, dass Digg die benötigten Informationen öffentlich verfügbar macht. In diesem Fall können Sie die benötigten Endpunkte unter **http://apidoc.digg.com/ListStories finden**.

Digg mit Python

Da Sie nun die Grundlagen der Digg-API kennen, können Sie eine URL eingeben, und im Gegenzug wird XML-Code zurück übergeben. Dies ist ein guter Start, aber nun kommen wir dazu, wie diese Informationen programmtechnisch verarbeitet werden. Als Erstes müssen wir eine URL abrufen, dies kann mit dem Python-Modul *urllib* erledigt werden. Öffnen Sie dazu eine Python-Shell.

```
>>> url='http://services.digg.com/stories/hot/?'
>>> appkey='http:///linuxformat.co.uk'
>>> import urllib
>>> diggargs ={ 'count': 1, 'appkey': appkey}
>>> foo=urllib.urlencode(diggargs)
>>> request = url+foo
>>> request 'http://services.digg.com/stories/hot/?count=1&
appkey=http%3A%2F%2Flinuxformat.co.uk'
>>>
```

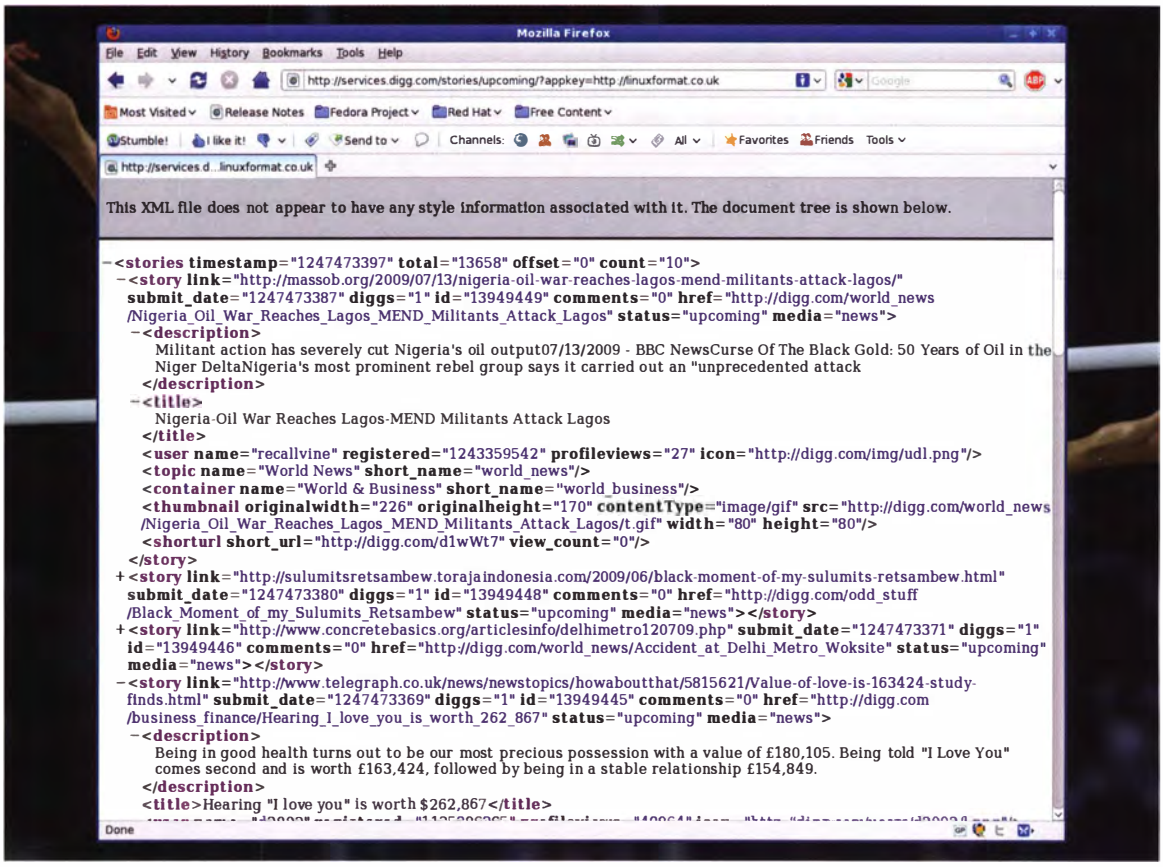
Als Erstes sollten Sie die URL und den Appkey für den späteren Gebrauch abspeichern. Wenn Sie das *urllib*-Modul importiert haben, wollen wir als nächstes einige Variablen an Digg übertragen. Beim Beispiel haben wir ein Python-Dictionary verwendet. Dies ist ein simples Konstrukt in geschweiften Klammern, welches **key=value**-Paare nimmt und in etwa wie eine Liste funktioniert. Erst kommt der Name des jeweiligen Wertes, gefolgt von einem Doppelpunkt und dem Wert der Variable. Die Werte können jeden gültigen Python-Wert besitzen, aber hier werden eher Strings oder Ganzzahlen verwendet.

Warum ein Dictionary?

Warum machen wir uns die Mühe, ein Dictionary aus unseren Argumenten zu erstellen? Dies hängt mit einer Hilfsfunktion des »

Quick-Tipp

Anfänger bleiben oft in der Python-Shell hängen, da man statt STRG+C das Terminal mit STRG+D verlassen muss.



› **Wenn Sie Firefox oder einen anderen Browser verwenden, wird die Anfrage in einer strukturierten Ansicht präsentiert.**

» *urllib*-Moduls zusammen, mit dessen Hilfe man aus den Argumenten automatisch einen Abfrage-String erstellen kann. Dies ist nicht so einfach wie das Verknüpfen von Strings, da HTML Regeln besitzt, welche Buchstaben als Anforderung gesendet werden können. Daher wird in Folge die Hilfsfunktion **urlib.urlencode** genutzt, um das Wörterbuch in Anfrage-Strings zu ändern.

Es wäre auch sehr fehleranfällig, wenn Änderungen anstehen und wir unsere Argumente für einen Anfrage-String nur konvertiert hätten. Die Anfrage wird gebildet, indem man die Basis-URL und den Anfrage-String kombiniert. Sie können auch nur einen Variablennamen eingeben, und Python zeigt den Wert an. Was bekommen wir also, wenn Sie sich an einem Server mit dieser Anfrage anmelden? Dies sollte so aussehen:

```
>>> response = urllib.urlopen(request)
>>> response.read()
'<?xml version="1.0" encoding="utf-8" ?>\n<stories
timestamp="1247411540" total="12257" offset="0"
count="1">\n <story link="http://www.thedailybeast.com/
blogs-and-stories/2009-07-11/young-gop-chooses-hate/"
submit_date="1247350492" diggs="109" id="13923829"
comments="46" href="http://digg.com/politics/Young_GOP_
Chooses_Hate" status="upcoming" media="news">\n
<description>Audra Shay, the Young Republican leader
accused of endorsing racism on Facebook, was elected head
of the group for GOP members under 40 this afternoon.</
description>\n <title>Young GOP Chooses Hate </title>\n
<user name="Pash1994" registered="1220537298"
profileviews="2218" icon="http://digg.com/users/Pash1994/l.
png" />\n <topic name="Political News" short_
name="politics" />\n <container name="World &amp;
Business" short_name="world_business" />\n <thumbnail
originalwidth="174" originalheight="174"
contentType="image/jpeg" src="http://digg.com/politics/
Young_GOP_Chooses_Hate/t.jpg" width="80" height="80"
/>\n <shorturl short_url="http://digg.com/d1wQDt" view_
count="309" />\n </story>\n</stories>'
```

Die **urlopen**-Funktion liefert ein dateiähnliches Format, welches wie andere Dateien auch verändert werden kann. Dies ist vor allem in Szenarios interessant, bei denen man sehr viele Daten erwartet. Der Befehl **response.read()** legt die Datei ab, und Sie können Sie in etwa so mit dem anderen Code kombinieren:

```
>>> response = urllib.urlopen(request).read()
```

Diesen Trick können Sie mit vielen Python-Objekten verwenden, allerdings macht es den Code etwas unübersichtlich. Sie erhalten durch diesen Trick eine große Menge an XML zurück. Um es als XML-Objekt einzufügen, müssen Sie wie folgt einige Methoden des Python-XML-Moduls verwenden:

```
>>> from xml.dom import xml.minidom
>>> x = minidom.parseString(response)
```

Daten von Objekten zu sammeln und ein strukturiertes XML-File daraus zu erstellen, ist im XML-Umfeld auch als „Marshalling" bekannt. Wir wollen allerdings genau das Gegenteil erreichen, nämlich ein Python-Objekt aus den Daten erstellen. Was nun folgt, ist vermutlich einer der am meisten kopierten Code-Abschnitte in der Welt von Python und XML.

```
class Bag: pass
def unmarshal(element):
    rc = Bag()
    if isinstance(element, minidom.Element):
      for key in element.attributes.keys():
        setattr(rc, key, element.attributes[key].value)
    childElements = [e for e in element.childNodes \
                     if isinstance(e, minidom.Element)]
    if childElements:
      for child in childElements:
        key = child.tagName
        if hasattr(rc, key):
            if type(getattr(rc, key)) <> type([]):
              setattr(rc, key, [getattr(rc, key)])
            setattr(rc, key, getattr(rc, key) + [unmarshal(child)])
        elif isinstance(child, minidom.Element) and \
              (child.tagName == 'Details'):
            # make the first Details element a key
            setattr(rc,key,[unmarshal(child)])
        else:
            setattr(rc, key, unmarshal(child))
    else:
      text = "".join([e.data for e in element.childNodes \
              if isinstance(e, minidom.Text)])
      setattr(rc, 'text', text)
    return rc
```

Ergänzend muss man dazu sagen, dass die **Bag**-Class zwar als eine Klasse ohne Inhalt definiert zu sein scheint, was in Python durchaus erlaubt ist, aber man kann eben auch im Vorfeld nicht wissen, welche (oder ob überhaupt) Daten in einer Abfrage enthalten sind. Dies ist ein gutes Beispiel für die Flexibilität von Python, da man quasi eine Klasse im Betrieb definiert.

Code-Anarchie?

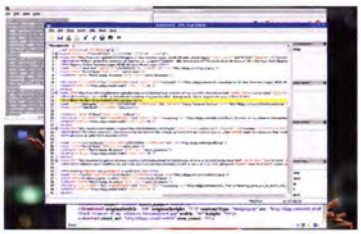
Die **unmarshal**-Funktion ist eine umgekehrte Prozedur, die jeden Knoten des XML DOM untersucht und daraus ein Python-Objekt erstellt. Wenn Sie ein API-Modul für Python erstellen wollen, welches den Digg-Output interpretiert, können Sie dies in einem strukturierteren Verfahren machen, weil Sie den Aufbau von XML kennen. Diese Methode ist eher ein globales Auslesen, mit dem Nachteil, dass das erstellte Objekt nicht wirklich zugänglich ist. Dies kann allerdings mit etwas Nachbearbeitung verbessert werden. Wir haben hier Story-Container innerhalb eines Containers mit dem Namen Storys – alle haben eigene Unterknoten für Kommentare, IDs und weitere Variablen. So schauen Sie sich den Story-Container näher an:

Quick-Tipp

Wenn Sie mit Python 3 experimentieren, werden Sie feststellen, dass *urllib* nicht mehr funktioniert. Diese wurde nämlich in *urllib.request*, *urllib.parse* und *urllib.error* aufgeteilt. Weitere Infos finden Sie unter: **http:// docs.python.org/library/urllib.htm**.

Mit XML experimentieren

In unseren Beispielen haben wir relativ übersichtliche XML-Elemente, die nicht zu sehr in die Tiefe gehen. Wenn Sie größere Dokumente bearbeiten oder eigene XML-Dateien erstellen wollen, benötigen Sie unbedingt einen guten Editor. Ein brauchbarer Editor ist beispielsweise der *XMLCopyEditor*, der Ihnen bei der Strukturierung helfen sollte. Auch Scans und die Suchfunktion sind hier besser umgesetzt.



› Ein XML-Editor kann Ihnen viel Zeit und Frust sparen.

```
>>> for item in bar.stories.story:
>>> print item.id, item.link, item.title.text
```

Es gibt natürlich auch keinen Grund, der dagegen spricht, weitere Programmteile zur Struktur hinzuzufügen. Wenn Sie beispielsweise wissen möchten, wo die Stories veröffentlicht wurden, können Sie dies mit der freien *GeoIP*-Library erreichen.

Mehr Daten hinzufügen

Das **GeoIP**-Modul ist bei den großen Distributoren erhältlich. Sie können es auch von MaxMind (**www.maxmind.com/app/python**) herunterladen. Die Funktion ist recht einfach – Sie müssen nur ein **GeoIP**-Objekt erstellen und dann die zur Verfügung stehenden Methoden nutzen, um über die IP-Adresse das jeweilige Land zu ermitteln.

```
>>> import GeoIP
>>> geo=GeoIP.new(GeoIP.GEOIP_STANDARD)
>>> geo.country_name_by_name('google.com')
'United States'
```

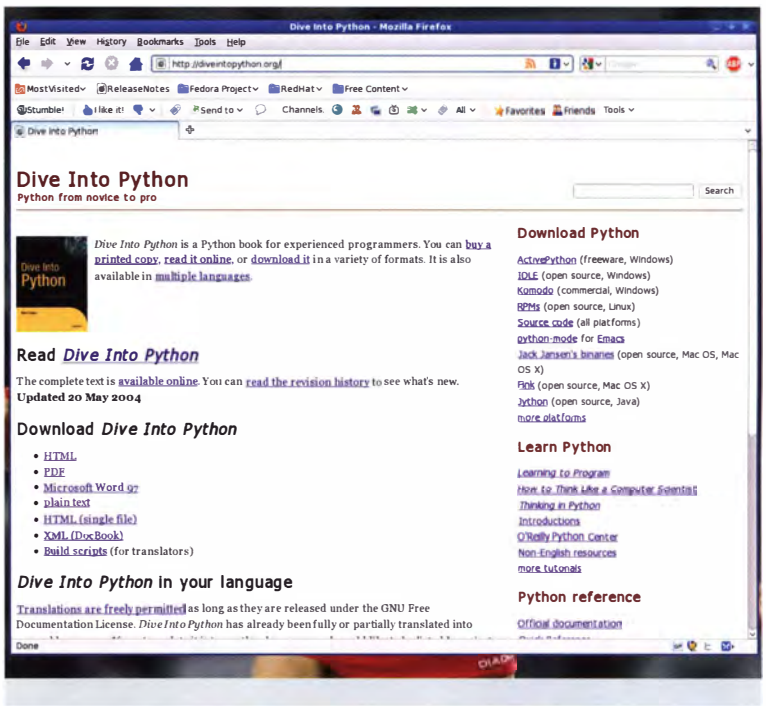
Der Code ist schlicht, hat aber auch die Einschränkung, dass man nur die Domain und nicht die URL verwenden kann. Mit dem Standard-Python-Modul **urlparse** kann man allerdings die gesamte URL verwerten: **www.python.org/doc/2.5.2/lib/module-urlparse.html**.

```
>>> import urlparse
>>> for item in bar.stories.story:
>>> item.country=geo.country_name_by_name(urlparse.
urlparse(item.link).netloc)
>>> print item.country
```

Hier haben wir unsere Schleife neu gestaltet und erstellen so ein neues Item **country**. Nun wird die Information der **urlparse**-Funktion an die **GeoIP**-Funktion weitergegeben und man erhält den Namen des jeweiligen Landes zurück. Wenn Sie einen Schritt weiter gehen wollen, können Sie die **country**-Werte nehmen und beispielsweise ein Dictionary der 100 beliebtesten Seiten erstellen. Sie müssen dazu nur ein leeres Dictionary erstellen und dann eins zum Wert des jeweiligen Landes hinzufügen, während man durch die Stories loopt. Das Dictionary kann anschließend in eine Liste übersetzt werden, und man kann sich den Inhalt grafisch anzeigen lassen.

```
#!/usr/bin/python
import urllib
from xml.dom import minidom
import urlparse, GeoIP, operator
url='http://services.digg.com/stories/hot/?'
appkey='http://linuxformat.co.uk'
geo=GeoIP.new(GeoIP.GEOIP_MEMORY_CACHE)
class Bag: pass
def unmarshal(element):
    rc = Bag()
    if isinstance(element, minidom.Element):
      for key in element.attributes.keys():
        setattr(rc, key, element.attributes[key].value)
     childElements = [e for e in element.childNodes \
            if isinstance(e, minidom.Element)]
    if childElements:
      for child in childElements:
          key = child.tagName
          if hasattr(rc, key):
            if type(getattr(rc, key)) <> type([]):
              setattr(rc, key, [getattr(rc, key)])
            setattr(rc, key, getattr(rc, key) + [unmarshal(child)])
          elif isinstance(child, minidom.Element) and \
            (child.tagName == 'Details'):
            # make the first Details element a key
            setattr(rc,key,[unmarshal(child)])
          else:
            setattr(rc, key, unmarshal(child))
    else:
      text = "".join([e.data for e in element.childNodes \
            if isinstance(e, minidom.Text)])
      setattr(rc, 'text', text)
    return rc
diggargs ={ 'count': 100, 'appkey': appkey}
foo=urllib.urlencode(diggargs)
request = url+foo
response = urllib.urlopen(request).read()
x = minidom.parseString(response)
bar = unmarshal(x)
hist = {}
for item in bar.stories.story:
    item.country=geo.country_name_by_name(urlparse.
urlparse(item.link).netloc)
    hist[item.country]=hist.get(item.country, 0) +1
sorted = sorted(hist.items(), key=operator.itemgetter(1),
reverse=True)
print sorted
```

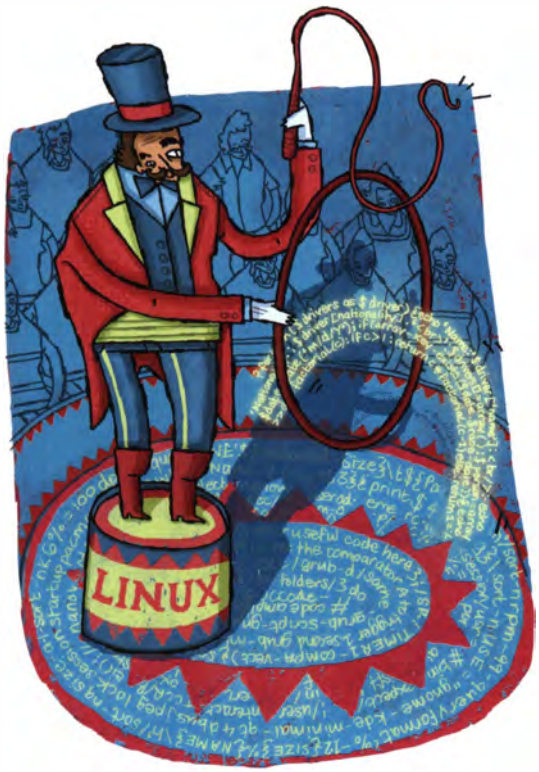
Im zurückliegenden Abschnitt haben wir viele Themenbereiche angeschnitten, auch wenn wir im Detail nur auf die Digg-Endpunkte eingegangen sind. Für eine umfangreichere API sollten Sie Klassen und Objekte verwenden, bei denen Stories, Anwender und weitere Details berücksichtigt werden. ■



› **Wenn Sie mehr Informationen zu Python suchen, sollten Sie sich Dive Into Python näher anschauen.**

Fotografie: Flickr-

Nick Veitch zeigt Ihnen, wie Sie Bilder von Ihrem Desktop direkt ins Internet hochladen.



Vor einiger Zeit dachte sich jemand, dass das Internet ja ganz gut für den passiven Konsum geeignet sei, aber dass es da doch noch etwas mehr geben sollte. Also wurde Folgendes beschlossen: Anstatt unsere kollektive Weisheit auf die Worte einiger bewanderter Propheten zu beschränken, die Websites betrieben, sollte jeder Schreibrechte für das Internet bekommen – **chmod ugo+rwx –R http://***, wenn Sie so wollen, also wirklich gefährlicher Stoff. Von diesem Moment an war es quasi unmöglich, ein vernünftiges Ergebnis zu bekommen, wenn man irgendetwas bei Google eintippte – und das selbst veröffentlichte Internet war geboren.

Bisher waren wir im Programmierteil dieses Bookazines größtenteils damit beschäftigt, Daten im Internet zu sammeln und sie für unsere dunklen Zwecke zu benutzen. In diesem Fall werden wir aber etwas zurückgeben. Die Cloud war gut zu uns, also werden wir sie mit ein paar Fotos füttern.

Es ist uns außerdem zu Ohren gekommen, dass die Konsole eine ziemlich elitäre Art sein soll, sich mit dem Web zu befassen. Verfechter der reinen Lehre sind womöglich der Meinung, dass es falsch sei, wenn jemand ohne Kenntnisse über drei verschachtelte Sets von Befehlszeilenoptionen etwas Sinnvolles erreichen kann. Wir werden aber hier diese Kluft schließen und eine grafische Benutzeroberfläche (GUI) bauen.

Python auf dem Desktop

Unsere Desktop-Anwendung wird sich sehr eng in das ihr zugrunde liegende System integrieren müssen. Da wir uns für irgendein System entscheiden müssen, nehmen wir die große Zahl der Gnome-Nutzer als Kriterium und wählen *GTK* für den GUI-Code. Falls Sie *KDE* verwenden, fühlen wir mit Ihnen. Allerdings schien *GTK* bei unseren vorbereitenden Experimenten die Aufgaben außerdem auch in weniger Zeilen zu bewältigen als *KDE/Qt*. Wenn Sie den Code für *wxWidgets* konvertieren wollen, sollte das nicht allzu schwierig sein, wenn Sie erst einmal die Funktionsweise verstanden haben. Bevor wir weitermachen, holen Sie zunächst *PyGTK* aus Ihrer Paketverwaltung.

Bisher haben wir Anwendungen Zeile für Zeile in der interaktiven Python-Konsole gebaut. Das ist bei der Erstellung einer Benutzeroberfläche jedoch schwieriger, da eine GUI weniger linear ist. Daher schreiten wir voran und gehen vom Schreiben eines Skripts zur Erstellung einer richtigen objektorientierten Anwendung über. Es wird manchmal diskutiert, was objektorientierte Programmierung ausmacht, aber was wir (zumindest hier) meinen, ist Folgendes: Wir werden ein Objekt erstellen, das unter bestimmten Bedingungen eine Aktion ausführt. Wenn wir nichts mit dem Objekt anstellen, tut es auch nichts für uns. Wenn wir es anstupsen, wird es das tun, was es tun soll, wenn es angestupst wird.

> „Die Cloud war gut zu uns, also werden wir sie mit ein paar Fotos füttern."

In diesem ersten Beispiel sollten Sie die nachfolgenden Befehle in den Editor Ihrer Wahl eintippen, statt sie Zeile für Zeile in den Interpreter einzugeben. Natürlich können Sie Letzteres trotzdem versuchen, doch wir raten davon ab, da wir mehrere Methoden einbauen, die nicht vor dem Ende aufgerufen werden. Dadurch ist es schwierig, Fehler aufzuspüren.

Unser erstes Objekt ist ein Anwendungsfenster, das auf dem Desktop geöffnet wird. Die meiste Zeit wird dieses Fenster nichts weiter tun, aber wenn Sie eine Datei in das Fenster ziehen, wird es Ihnen deren Pfadnamen geben – der erste Schritt

Flickr-Programmierschnittstelle

Bei Flickr geht es nicht bloß um das Hochladen und Kommentieren von Bildern. Dank Exif, Tags und vielen anderen Datentypen ist Flickr auch ein riesiger Spielplatz für die Erstellung von Apps. Eine der tollsten Eigenschaften der Flickr-Programmierschnittstelle ist, wie gut sie dokumentiert ist. Nicht nur wird jede Aufforderung erklärt, sondern es finden sich auch jede Menge Beispiele in der Online-Dokumentation, und vieles können Sie sogar über Webformulare ausprobieren. Alle Infos finden Sie unter **flickr.com/services/api**.

Uploader

bei der Erstellung unseres selbstgemachten Flickr-Clients. Wir müssen zuerst die relevanten Module laden und eine Methode definieren, die den Dateinamen auf dem Bildschirm erscheinen lässt. Dann erstellen wir eine *GTK*-Anwendung. Hier ist der dafür benötigte Code:

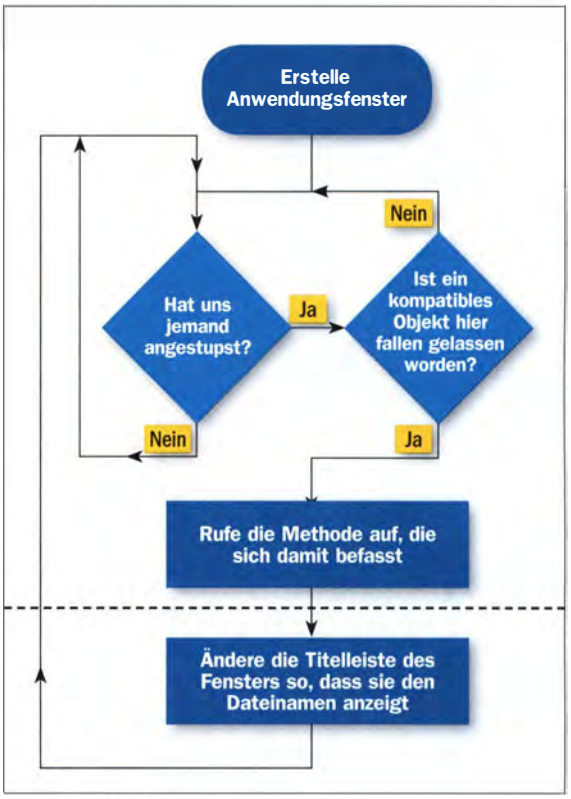
```
import pygtk
import gtk
def rec_cb(widg, context, x, y, selection, info, time):
    #received a valid object
    filename= selection.data[7:-2]
    print filename
    l.set_text(filename)
#gtk app-building magic
w = gtk.Window()
w.set_size_request(200, 200)
w.drag_dest_set(gtk.DEST_DEFAULT_MOTION | \
        gtk.DEST_DEFAULT_HIGHLIGHT |\
        gtk.DEST_DEFAULT_DROP \
        , [("UTF8_STRING", 0, 0 )], \
        gtk.gdk.ACTION_COPY)
w.connect('drag_data_received', rec_cb)
w.connect('destroy', lambda w: gtk.main_quit())
l = gtk.Label()
w.add(l)
w.show_all()
gtk.main()
```

Bevor wir uns den Inhalt des Codes näher ansehen, führen Sie ihn bitte erst einmal aus: Es sollte sich ein Fenster öffnen. Ziehen Sie nun eine Datei vom Desktop über das Fenster, bis es hervorgehoben wird. Lassen Sie die Datei los, und der Dateiname erscheint in der Titelleiste.

Das nebenstehende Diagramm verdeutlicht den Prozess: Das Fenster dient als Zielort von Drag-and-Drop-Ereignissen, und wenn etwas im Bereich des Fensters fallen gelassen wird, ruft die Anwendung aus dem Code die für die Bearbeitung zuständige Prozedur auf. In diesem Tutorial beschäftigen wir uns nur mit dem Teil des Prozesses, der sich unterhalb der gestrichelten Linie im Diagramm abspielt. Um den Rest kümmert sich die *GTK*-Anwendung.

Tiefer in GTK einsteigen

Sie werden es leichter haben, wenn Sie zuerst die Mechanismen von *GTK* verstehen, bevor Sie sich mit *PyGTK* beschäftigen. Das Problem ist, dass alle Beispiele in C++ und nicht in Python sind, aber ein Mix von Hintergrundmaterial ist trotzdem nützlich. Über *PyGTK* gibt es einen guten, wenn auch etwas älteren Guide (nur in englischer Sorache) unter **pygtk.org/pygtk2tutorial**. Für einen generelleren Überblick über Gnome/*GTK* empfehlen wir den *Official Gnome 2 Developer's Guide* (ISBN 978-1593270308). Wenn Sie es sich beim Erstellen von Interfaces etwas einfacher machen wollen (und in Kauf nehmen, einen Code zu generieren, den Sie nicht verstehen), dann schnappen Sie sich *Glade* in der Paketverwaltung und spielen Sie ein bisschen mit den Widgets herum.

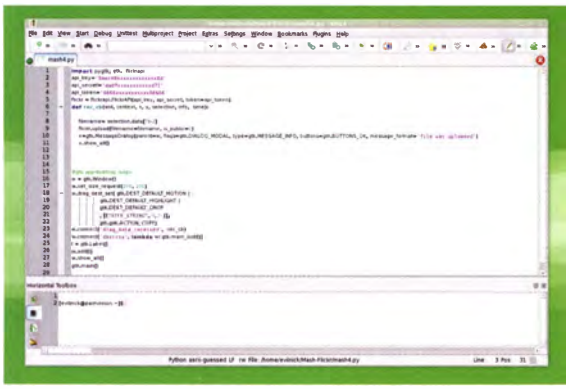


› **Das ist die Programmlogik unserer App. Zum Glück müssen wir nur einen kleinen Teil des Codes selber schreiben.**

So funktioniert der Code

Ignorieren Sie einstweilen die Funktionsdefinition am Anfang des Scripts, und wenden Sie sich dem Teil zu, der mit der Zeile **w = gtk.Window()** beginnt. Die Zeile erstellt unser Anwendungsfenster-Objekt. Die nächste Zeile ist nicht nur die wichtigste, sondern auch die vielleicht verwirrendste. Wir haben die Zeile aufgeteilt (indem wir den „\"-Fortsetzungsmarker für Python verwendet haben), damit Sie sehen können, dass das Fenster eine **drag_dest_set**-Methode mit drei Parametern aufruft. Der erste besteht aus einigen Flags und sagt *GTK*, wie sich das Fenster verhalten soll. Über das Flag **DEFAULT_MOTION** legen wir fest, dass wir keine Aktionen selbst bearbeiten wollen. Das Flag **DEFAULT_HIGHLIGHT** bedeutet, dass der Drop-Bereich automatisch hervorgehoben wird, wenn sich ein kompatibles Objekt über dem Fenster befindet. **DEFAULT_DROP** bedeutet, dass die Daten, die vom Objekt übermittelt werden, automatisch angenommen werden und ein **drag_data_received**-Signal erzeugt wird.

Der nächste Parameter ist eine Tupel-Liste. In unserer Liste haben wir nur ein Tupel. Jedes Tupel legt einen akzeptierten Datentyp fest (**UTF8_STRING** in unserem Fall), beginnend mit einem String, der den Typ enthält, gefolgt von einigen Flags, die die Datenbenutzung auf dieselbe Anwendung oder »



› Nur einige wenige Codezeilen liefern eine GUI, aber versuchen Sie zu verstehen, wie diese funktionieren.

» Widgetstruktur beschränken können, und einem Identifikator. Letzterer ist nützlich, wenn Sie viele Datentypen akzeptieren. Der letzte Parameter bestimmt den Typ der akzeptierten Aktionen, nochmals über ein im *GTK*-Modul definiertes Flag. In unserem Fall benutzen wir ein Flag für eine **copy**-Aktion, da sonst die Originaldaten zerstört würden.

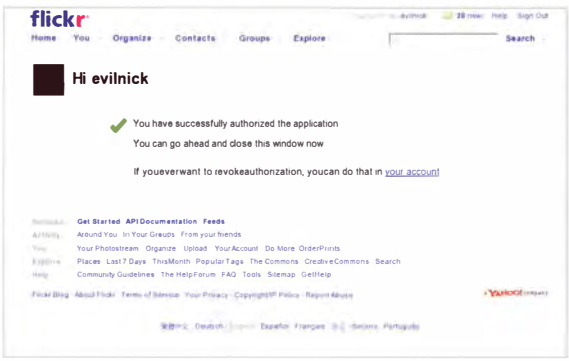
Wo kam nun dieses **UTF8_STRING** her? Nun, der andere Teil des Codes, den Sie nicht sehen, ist der Desktop an sich. Der Gnome-Desktop benutzt ständig *GTK*-Aktionen für Drag-and-Drop-Funktionen – vor allem für die Platzierung von Dateien in Ordnern. Das bedeutet, dass Ihre Desktop-Icons bereits als Quellen für Drag-and-Drop-Operationen festgelegt sind und über eine bestimmte Anzahl an möglichen Zieltypen verfügen. Wenn Sie diese aufgelistet bekommen möchten, ändern Sie den Code und ergänzen Sie **l.set_text(filename)** mit Folgendem:

```
l.set_text('\n'.join([str(t) for t in context.targets]))
```

Wir benutzen das UTF8-Format, weil wir den Namen des Objekts für den späteren Gebrauch festhalten wollen, und weil Python UTF-Strings mag.

Jetzt haben wir eingestellt, was das Fenster annehmen wird. Die nächste Zeile verbindet das **drag_data_received**-Signal mit der **drag_dest_set**-Methode, die wir zuvor definiert haben. Wir haben sie in der Zeile davor festgelegt, sodass dieses Signal gesendet wird, wenn etwas über unserem Fenster gedroppt wird. Wenn wir beides nicht verbinden, wird allerdings auch nichts passieren.

Die darauffolgende Zeile legt die Exit-Routine für das **destroy**-Signal fest, das gesendet wird, wenn der Benutzer auf den Schließen-Button des Fensters klickt. Danach fügen wir ein Label-Widget hinzu (um den Text darzustellen) und rufen die **show**-Methode für unser Fenster auf, die alle Elemente einfriert und *GTK* signalisiert, dass das Objekt benutzt werden kann. Es wird zwar nicht dargestellt, bis die Anwendungsschleife aufgerufen wird, aber dies ist die nächste Zeile. Wenn **gtk.main()** aufgerufen wird, übernimmt *GTK* praktisch die Kontrolle über unser Programm. Alle danach ausgeführten Aktionen (Fenster verschieben, Fenstergröße ändern, Fenster schließen usw.) werden vom *GTK*-Code ausgeführt – mit Ausnahme der Funktion, die wir unserem **data received**-Signal angehängt haben. Diese Methode ist als Rückruffunktion (engl. „callback“) bekannt, denn sie wird von der Hauptanwendung aufgerufen, sobald eine Aktion stattfindet – in diesem Fall unser Signal.



› Sie (oder Ihre Anwender) müssen sich bei Flickr einloggen und die App einmalig autorisieren. Das Flickr-API-Modul wird die Zugangsinformationen allerdings speichern.

Unser Code wird nun hoffentlich allmählich deutlicher. Das generierte Signal für die Aktion bringt jede Menge Informationen mit sich, die uns nicht interessieren. Wir bekommen allerdings auch einen Identifikator für das Widget, das das Signal bekommen hat, sowie Kontextinformationen und Daten über dessen Position, über die Auswahl und darüber, was für eine Datei es ist und wann sie gedroppt wurde. Die erhaltenen Daten sind ein Teil unseres Objekts, sodass wir sie extrahieren und dann die ersten sieben sowie die letzten zwei Zeichen abschneiden können, die ein **file://** und den Marker für das Ende der Zeile enthalten. Dadurch erhalten wir den Pfadnamen, den wir verwenden können, um die Daten für das Label in unserem Fenster zu überschreiben, indem wir die relevante Methode für unser Objekt aufrufen, **l**.

Die Daten benutzen

Einen GUI-Code zu erklären, nimmt viel Platz in Anspruch, aber wenn Sie Codes wie diesen entwickeln möchten, ist es wichtig, dass Sie verstehen, wie er funktioniert. Wir haben jetzt also einen Dateinamen und können etwas damit anfangen. Um die Datei bei Flickr hochzuladen, müssen Sie zuerst einige Dinge erledigen. Zu Beginn brauchen Sie einen Flickr-Account – erstellen Sie sich einen auf **flickr.com**, falls Sie noch keinen haben. Das Zweite, was Sie benötigen, ist ein API-Schlüssel samt Secret für die Programmierschnittstelle. Diesen bekommen Sie, wenn Sie sich als nicht-kommerzieller User auf **flickr.com/services/api** über den Link „API-Schlüssel“ registrieren. Jetzt brauchen Sie Zugriffsdaten, die der Anwendung Zugang zu einem speziellen Flickr-Account gewährt. So loggt sich die Anwendung in Flickr ein:

```
>>> import flickrapi
>>> api_secret='xxxxyoursecretxxxx'
>>> api_key='yyyyyyyyourkeyyyyyyy'
>>> flickr=flickrapi.FlickrAPI(api_key,api_secret)
```

Andere GUIs

Python ist nicht darauf eingeschränkt, *GTK* zu benutzen – im Gegenteil, es gibt viele Werkzeuge zur Bildung eines GUI. Wir empfehlen Ihnen *PyQt* und *PyKDE* als Alternativen – vor allem, wenn Sie ein Interface mit dem KDE-Desktop haben möchten. *PyKDE* ist *PyQt* sehr ähnlich, obwohl es auch einige Unterschiede gibt. KDE verwendet das *Qt*-Toolkit ausgiebig, aber alle Objekte und Methoden sind in einer Extraebene eingepackt, daher auch die Existenz beider Anwendungen – *PyKDE* für KDE-Apps und *PyQt* für Standard-*Qt*-Apps.
Eine weitere Alternative ist *wxWidgets*. Abgesehen davon, dass das Programm sehr einfach und leicht zu bedienen ist, liegt der Hauptvorteil darin, dass *wxWidgets* plattformübergreifend ist und die Bibliotheken zu nativen Toolkits verlinken. Dadurch sieht eine *wxWidgets*-App unter Linux wie eine Gnome-/*GTK*-App, unter Windows wie eine Standard-Windows-Anwendung und auf einem Mac wie ein Standard-OS-X-Programm aus.

```
>>> (token,frob)= flickr.get_token_part_one(perms='write')
>>> flickr.get_token_part_two((token, frob))
u'7215780808080-a94e70effffeebb01'
```

Wenn Sie zur vorletzten Zeile kommen, passiert etwas Verrücktes: Ihr Standard-Browser öffnet sich und navigiert zu Flickr. Dort müssen Sie sich einloggen und ein paar Fragen beantworten, um der App zu erlauben, Flickr zu benutzen. Vergessen Sie aber nicht, noch die letzte Zeile einzutippen, da diese Ihnen die Daten liefert, mit denen Sie Ihren Flickr-Account und die Anwendung verbinden. Wir benutzen die Daten, damit sich die Anwendung für Sie einloggen kann, aber Sie können das auch in der App erledigen, wenn Sie möchten. Das **flickrapi**-Modul speichert die Daten ohnehin standardmäßig, sodass Sie die App nur einmal autorisieren müssen. Wenn Sie nicht möchten, dass die Zugangsdaten gespeichert werden, können Sie die Option aber auch ausschalten.

Endlich ist es an der Zeit, etwas hochzuladen. Das **flickr**-Objekt, das wir vom Modul aus erstellen, hat eine Methode zum Hochladen von Bildern. Hier ist die komplette Liste von Parametern, die es akzeptiert:

» **filename**: Der Dateiname des Bildes. Dies ist der einzige zwingend erforderliche Parameter.
» **title**: Der Titel des Fotos.
» **description**: Der Beschreibungstext.
» **tags**: Ein String mit einer Liste von Tags, jeweils durch Leerzeichen getrennt.
» **is_public**: Hier sollte eine **1** stehen, wenn das Foto öffentlich ist, und eine **0**, wenn es privat ist. Der Standard ist „öffentlich".
» **is_family**: Hier sollte eine **1** stehen, wenn ein privates Foto für die Familie sichtbar sein soll, und eine **0**, wenn nicht. Der Standard ist „nicht sichtbar".
» **is_friend**: Hier sollte eine **1** stehen, wenn ein privates Foto für Freunde sichtbar sein soll, und eine **0**, wenn nicht. Der Standard ist „nicht sichtbar".
» **callback**: Eine Methode, die zwei Parameter – **progress** und **done** – erhält.
» **format**: Das Antwortformat.

Zum Glück ist nur der Dateiname zwingend erforderlich. Der Rest der Parameter ist optional und kann über eine **key=value**-Methode eingefügt werden. Hier ein Beispiel:

```
MyFlickrObject.upload(filename='/home/evilnick/
plop.jpg', tags='plop image jpeg', title='This is Plop!')
```

Wir müssen nicht alle Parameter verwenden oder uns auch nur merken, in welcher Reihenfolge diese angeführt werden – es könnte nicht einfacher sein. Wenn Sie sich mit Flickr auskennen, sind Ihnen die Tags und die Privatsphäre-Optionen vermutlich schon vertraut. Die einzigen Elemente, für die eine Erklärung nötig ist, sind die Optionen **callback** und **format**. Letzteres ist lediglich dazu da, dem Flickr-Server mitzuteilen, wie Sie die Antwortdaten formatiert haben möchten. Die **callback**-Option erlaubt Ihnen, eine Methode in Ihrem Code einzubauen, die Informationen aus dem **upload**-Prozess gewinnt und Ihnen sagt, wie weit dieser fortgeschritten ist oder wann er beendet wurde. Wenn wir den Benutzern unserer App etwas Feedback geben möchten, können wir diese Option benutzen.

Nach einigen Änderungen für den Upload sieht unser Code so aus:

```
import pygtk, gtk, flickrapi
api_key='--your-key--'
api_secret='--your-secret--'
api_token='--your-token--'
flickr = flickrapi.FlickrAPI(api_key, api_secret, token=api_token)

def rec_cb(wid, context, x, y, selection, info, time):
    filename= selection.data[7:-2]
    flickr.upload(filename=filename, is_public=0)
    x=gtk.MessageDialog(parent=w, flags=gtk.DIALOG_MODAL,
    type=gtk.MESSAGE_INFO, buttons=gtk.BUTTONS_OK,
    message_format='file was uploaded') x.show_all()

#gtk app-building magic
w = gtk.Window()
w.set_size_request(200, 200)
w.drag_dest_set(gtk.DEST_DEFAULT_MOTION | \
        gtk.DEST_DEFAULT_HIGHLIGHT |\
        gtk.DEST_DEFAULT_DROP \
        , [("UTF8_STRING", 0, 0 )], \
        gtk.gdk.ACTION_COPY)
w.connect('drag_data_received', rec_cb)
w.connect('destroy', lambda w: gtk.main_quit())
l = gtk.Label()
w.add(l)
w.show_all()
gtk.main()
```

Hier sehen wir: Das Einzige, was wir abgesehen vom Setup-Code an Flickr-Arbeit machen müssen, ist es, die **upload**-Methode im Rückruf für die Drag-and-Drop-Aktion aufzurufen. Um den Benutzer wissen zu lassen, dass der Upload funktioniert hat, lassen wir ein Fenster aufpoppen, das ihm dieses anzeigt.

Fertig! Ein Desktop-Drag-and-Drop-Uploader in weniger als 30 Code-Zeilen – vielleicht ist GUI-Programmierung ja gar nicht so schwer.

Ausblick

Wenn Sie diese Anwendung erweitern möchten, könnten Sie Umschalt-Buttons für die verschiedenen Privatsphäre-Flags hinzufügen oder ein Widget zur Texteingabe entwickeln, das Ihnen erlaubt, einen Titel, eine Beschreibung oder Tags festzulegen. Denkbar wäre auch der Einbau einer Miniaturansicht des Bildes im Drop-Bereich oder das Hinzufügen eines Fortschrittsbalkens, der mittels Rückruf von der **flickr.upload()**-Methode aktualisiert wird. All diese Erweiterungen erfordern allerdings vertiefte Kenntnisse über *GTK* und *PyGTK*. (siehe Kasten auf Seite 107). ■

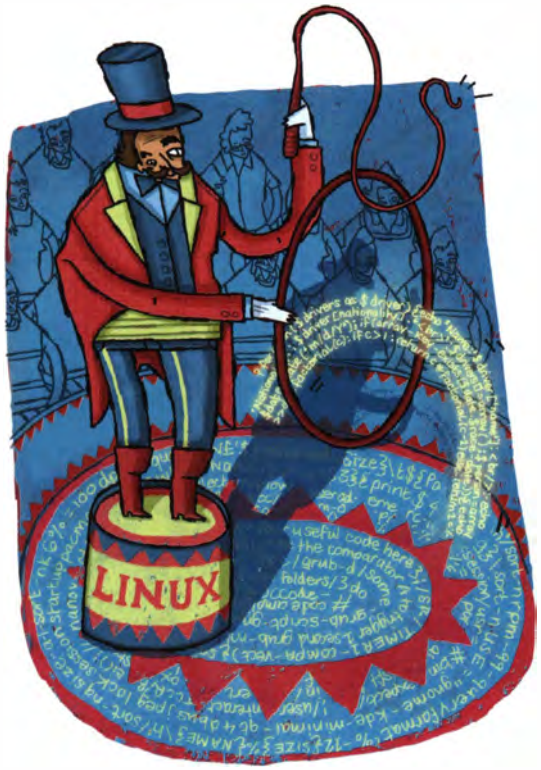
› **Es ist immer schön, Feedback zu bekommen – auch wenn es nur eine Standard-Dialogbox ist. Sie können den Code jedoch erweitern und für mehr Benutzerfreundlichkeit sorgen.**

Quick-Tipp

Die GTK-Dokumentation ist etwas ausführlicher als die für PyGTK und kann hilfreicher sein, wenn Sie nicht weiterkommen. Sie finden sie unter **developer.gnome.org/gtk-tutorial**.

Karten: WOEIDs

Mit diesem Guide erhalten Sie Einblick in die Verwendung von Geo-Daten im Maps-Kontext. So finden Sie sehenswerte Fotolocations.



Stellen Sie sich vor, dass eine Freundin eine Reise nach Barcelona antreten will. Natürlich werden auf so einer Reise auch viele Fotos geschossen. Nun werden Sie um Rat gefragt, welche Ecken in Barcelona gute Motive für Fotos abgeben würden. Abseits der bekannten Fleckchen wollen Sie natürlich auch mit einigen außergewöhnlichen Ortsvorschlägen glänzen. Sie brauchen also eine Foto-Karte. Viele Kameras und Smartphones versehen Fotos bereits auf Wunsch mit Geo-Tags (in den Foto-Exif-Daten werden unter anderem Längen- und Breitengrade verborgen). Fotoseiten wie beispielsweise Flickr nutzen solche Daten und verarbeiten diese weiter. So kann man durchaus auch alternative Fotolocations finden.

WOEIDs zeigen den Weg

Der Yahoo-Gelocation-Dienst ist für Entwickler zugänglich. Eine der wichtigsten Funktionen bei diesem Dienst ist die Zuordnung von eindeutigen IDs, die für reale Orte und Regionen stehen. So gibt es beispielsweise eine eindeutige ID, die für den Eiffelturm in Paris steht. Diese ID ist darüber hinaus mit anderen IDs in der Nähe verknüpft. Diese Orts-IDs werden als WOEIDs („Where On Earth IDs“) bezeichnet und werden auch bei anderen Lokations-Anwendungen wie beispielsweise Flickr verwendet. Wenn Sie mehr zum Thema lesen möchten, finden Sie unter **http://tinyurl.com/yl7xz9g** weitere Informationen. Für unsere Zwecke ist es jetzt nur wichtig, dass Städte eigene WOEIDs besitzen und diese sich in weitere WOEIDs verzweigen.

Der WOEID-Dienst von Yahoo benötigt einen API-Key, damit Sie sich damit befassen und diesen nutzen können. Bei Flickr können Sie die API-Sandbox nutzen und eine Anfrage stellen und auf diese Weise etwas einfacher an die gesuchte WOEID gelangen. Gehen Sie auf **http://tinyurl.com/ykbnmgb**, und geben Sie als Suchbegriff „Barcelona“ ein. Die ausgegebenen Daten zeigen, dass Barcelona mit der Kennzahl 753692 versehen ist. Die Flickr-API bietet mit **places.getChildrenWithPhotosPublic()** eine weitere brauchbare Methode. Diese Methode gibt eine Datenauflistung von zugehörigen IDs an, die mit der gelieferten ID interagieren. Alle IDs werden mit Längen- und Breitengraden angegeben, und darüber hinaus gibt es Hinweise auf interessante Plätze innerhalb dieses ID-Bereichs. Um die Flickr-API abfragen zu können, benötigen Sie einen registrierten Flickr-Account und einen gültigen API-Key (siehe Seite 113). Sie benötigen ebenfalls das Flickr-API-Modul für Python. Sie müssen also das zugehörige Paket für Ihre Distribution holen oder unter **http://stuvel.eu/projects/flickrapi** nachschauen. Wenn die Vorarbeiten abgeschlossen sind, können wir nun testen, wie die WOEID-Bildersuche funktioniert:

```
>>>import flickrapi
>>>apikey='bxxxxxxxxxx2'
>>>woeid=753692
>>>flickr = flickrapi.FlickrAPI(apikey)
>>>request=flickr.places_getChildrenWithPhotosPublic
(woe_id=woeid)
>>>list=request[0].getchildren()
>>>for item in list:
...         if item.attrib['place_type_id'] == '22':
...       print item.attrib['latitude'], item.attrib['longitude']
...       print item.text
...
41.380 2.176
Ciutat Vella, Barcelona, Catalonia, ES, Spain
41.393 2.161
L'example, Barcelona, Catalonia, ES, Spain
41.383 2.183
Barceloneta, Barcelona, Catalonia, ES, Spain
41.386 2.177
Ribera, Barcelona, Catalonia, ES, Spain
41.406 2.179
La Dreta De L'eixample, Barcelona, Catalonia, ES, Spain
41.379 2.167
...
```

Wir haben die **getChildrenWithPublicPhotos**-Methode verwendet, die uns ein Ergebnis geliefert hat. Wie Sie bei der Flickr-Abfrage gesehen haben, erhalten Sie so ein Objekt mit einem Element **places**, welches selbst eine Liste mit weiteren Plätzen enthält. Mit der **.getChildren**-Methode werden die Informationen aufgefächert. Der zugehörige **place_type** der Daten ist eine Nachbarschaft „type 22“. Der Grund dafür liegt in

im Detail

› **OpenStreetMaps ist die beste Quelle für freie Map-Daten, mit denen Sie plotten und publishen können.**

überlappenden Nachbarschaften begründet, die unsere Daten ansonsten mit Mehrfacheinträgen verfälschen würden. Die **.attrib()**-Methode gibt die genannten Schlüsselwerte für die individuellen Objekte aus, und der Beschreibungstext liegt im **.text**-Format vor. Die anderen Werte beinhalten **photo_count** und die WOEIDs. Mit den Werten für die bekannten Punkte können wir diese einfach auf eine Karte übertragen. Natürlich sind so auch Diagramme mit den beliebtesten Orten möglich. Nun brauchen Sie nur noch eine schöne Grafik mit den vorliegenden Daten zu erstellen.

Einen Graphen plotten

Immer wenn Konturen dargestellt, Graphen gezeichnet oder Punkte vermerkt werden sollen, greifen Python-Anwender zuerst zur *Matplotlib*-Bibliothek. Ähnlich wie ihre proprietäre Schwester *Matlab* ist *Matplotlib* eine Bibliothek von Funktionen, welche Daten in grafisch ansehnliche Konstrukte umsetzen. Auch dieses Python-Modul ist üblicherweise für Ihre Distribution erhältlich. Idealerweise sollten Sie auch das *Numpy*-Modul installieren. Dieses rüstet Python mit nützlichen Features und Funktionen für diese Art von Arbeit aus. *Matplotlib* besitzt eine nützliche Funktion namens **contourf**, mit der man **x**-, **y**- und **z**-Werte nehmen und damit ein grafisches Gebilde erstellen kann, das durch die Highlights um die hohen Werte herum praktisch eine Heatmap darstellt.

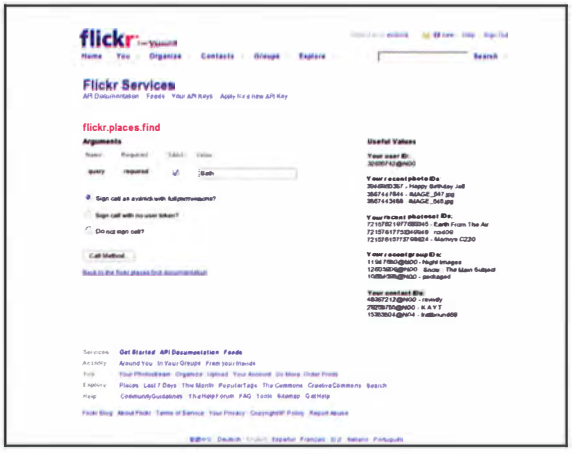
Bevor wir aber unsere Daten verwenden können, gibt es ein Problem: die **contourf**-Funktion arbeitet nur mit einem regulären Datenraster – unsere Punkte und Werte sind hingegen zufällig verteilt. Sie müssen daher ein leeres Raster erstellen und unsere Punkte dort eintragen, indem Sie eine Art „nächste Nachbarschaft"-Interpolation anwenden. Die **numpy.linspace()**-Methode erzeugt eine Liste mit Werten innerhalb definierter Grenzen. Sie können die **griddata**-Funktion von *Matplotlib* verwenden, um unsere Punkte auf einem normalen Raster zu interpolieren. Der Code dazu sieht so aus:

```
for item in list:
    if item.attrib['place_type_id'] == '22':
            y.append(float(item.attrib['latitude']))
            x.append(float(item.attrib['longitude']))
            z.append(int(item.attrib['photo_count']))
xmin=min(x)
xmax=max(x)
ymin=min(y)
ymax=max(y)
x += xmin, xmin, xmax, xmax
y += ymin, ymax, ymin, ymax
z += 0, 0, 0, 0
yrange=ymax-ymin
xrange=xmax-xmin
maxrange=max(xrange, yrange)
stepsize=maxrange/500
# create empty grid
xi = numpy.linspace(xmin,xmax,int(xrange/stepsize))
yi = numpy.linspace(ymin,ymax,int(yrange/stepsize))
#grid the data
zi = griddata(x,y,z,xi,yi)
```

Wir haben uns hier auch um andere Dinge gekümmert. Durch die maximalen und minimalen Werte für **x** und **y** haben wir die Grenzen für die Karte dieser Umgebung gesetzt. Sie können diese Punkte zum Datensatz hinzufügen, damit die Konturen auch bis zum Rand der Karte reichen und nicht abgeschnitten werden. Sie können mit diesen Werten auch die Schrittgröße bei diesem Raster berechnen. Bliebe nur noch die **contourf**-Funktion zum Anzeigen des Gradienten in diesem Bereich:

```
CS = plt.contour(xi,yi,zi,15,linewidths=0.5,colors='k')
CS = plt.contourf(xi,yi,zi,15,cmap=plt.cm.jet)
plt.colorbar() # draw colorbar
# plot data points.
plt.scatter(x,y,marker='o',c='b',s=5)
plt.xlim(xmin,xmax)
plt.ylim(ymin,ymax)
plt.title('Barcelona')
plt.savefig('barcac.svg')
plt.show()
```

»



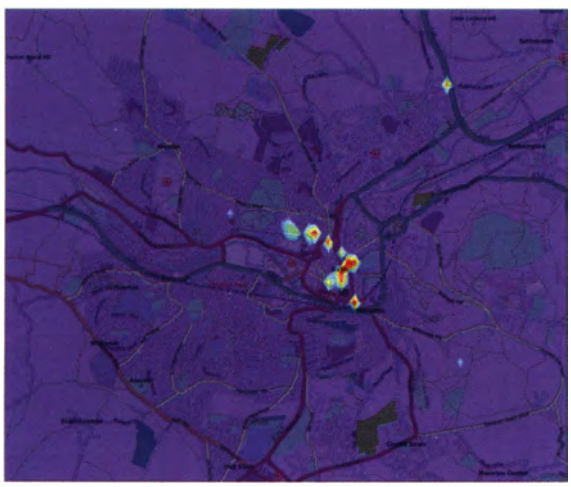
› **Flickr hat nicht nur eine sehr gute API-Dokumentation, sondern es ist auch möglich, über die Webseite die API abzufragen.**

» Hier haben wir die Konturlinie mit einer Dicke von 0,5 Millimetern in Schwarz („**k**“) geplottet, gefolgt von der ausgefüllten Kontur-Methode. Der **cmap**-Parameter ist eine der vordefinierten „Colormaps“ für **Matplotlib**, mit der wir eine gute Farbauswahl von Blau über Gelb bis Rot zur Verfügung haben. Nun müssen nur noch die erzeugten Bilder auf einer Karte dargestellt werden. Sie könnten Google Maps nehmen, doch die Karten gehören Google, daher empfehlen wir OpenStreetMap als Alternative. Es gibt keine direkte Methode, mit der man ein PNG aus einer definierten Box auslesen kann, aber Sie können eins manuell herunterladen, indem Sie auf dem Export-Tab auf der Mainpage die Längen- und Breitengerade eingeben. Nutzen Sie den OSM-Renderer auf der höchsten Zoomstufe. Ihre Map-Bilder und die gerenderte Heatmap können in *Gimp*, *Scribus* oder *Inkscape* kombiniert werden. Bei *Inkscape* können Sie eine SVG-Version der Konturkarte laden, die Dursichtigkeit auf 50 % setzen und das Bild einfach über die gerenderte Straßenkarte ausdehnen. Die Maßstäbe unserer Heatmap sind allerdings nicht ganz korrekt, denn unsere **x**- und **y**-Koordinaten weichen von der Einteilung eines Planeten in Längen- und Breitengrade ab – die Ungenauigkeit ist bei kleinen Karten allerdings vernachlässigbar. Allerdings müssen Sie dennoch relativ sorgfältig die Konturen anpassen, damit die Karte so genau wie möglich wird.

Weitere Ansätze

Unser Ansatz funktioniert nur für größere Gebiete, bei denen genügend Informationen zur Nachbarschaft vorliegen. Um eine akkuratere Ansicht zu bekommen oder um eine Gegend zu plotten, für die wenig bis gar keine Daten vorliegen, muss man anders an die Sache herangehen. Die „Brute Force“-Methode bestünde in dem manuellen Herunterladen aller Fotos, die in der in Frage kommenden Gegend zur Verfügung stehen. Dies wollen Sie gewiss nicht bei größeren Städten anwenden, aber für kleine Gebiete ist diese Methode durchaus brauchbar. Die Flickr-Standardmethode **photos.search()** bietet nur eine Möglichkeit an, mit der man Informationen von einer bestimmten WOEID auslesen kann. Die Bildinformationen werden in einem speziellen Listenformat zurückgegeben. Da das Arbeiten auf den Flickr-Servern zeitraubend ist, sollten Sie die heruntergeladenen Daten sammeln, damit diese später auch lokal verfügbar sind. Die Datenvorbehandlung gibt Ihnen auch die Möglichkeit, Bilder mit falschen Informationen oder ungenauen Beschreibungen zu entfernen.

```
import flickrapi
apikey='bxxxxxxxx52' #your key here
```



› **Hier können Sie sehen, wo Menschen Bilder aufnehmen und somit vermutlich die interessantesten Orte für Touristen zu finden sind.**

› **In Tampa in Florida scheint es nicht so viele Fotomotive zu geben.**

```
out=open("bath.csv","w")
woeid=12056
geolist=[]
flickr = flickrapi.FlickrAPI(apikey)
#request=flickr.interestingness_getList(extras='geo',
per_page=500)
request = flickr.photos_search(woe_id=woeid, per_page=250)
foo=request.getchildren()
pages=int(foo[0].items()[3][1])
print pages
for i in range(1, pages+1):
    try:
        request = flickr.photos_search(woe_id=woeid,
extras='geo',per_page=250, page=i)
        foo=request.getchildren()
        bar=foo[0].getchildren()
        #bar is now list of photoelements
        print 'attempt '+str(i)
        for pic in bar:
            if (pic.attrib['accuracy'])>14:
                lat=(float(pic.attrib['latitude']))
                lon=(float(pic.attrib['longitude']))
                if (lat, lon)!=(0.0, 0.0):
                    out.writelines(str(lat)+', '+str(lon) +'\n')
except:
        print 'skipped ' + str(i)
out.close
```

Dieses Skript loopt durch alle Ergebnisseiten, um Daten auszulesen, und speichert diese dann in einer CSV-Datei mit Längen- und Breitengraden. Sie können diese Daten für eine genauere Karte verwenden – dieses Mal allerdings ohne z-Achse. Sie haben nur Punkte, deren Wert gleich ist. Wenn Sie also den gleichen Prozess wie zuvor verwenden, kommt ein flaches Bild mit nur einem Punkt heraus. Die Daten müssen zuerst gruppiert werden („Binning“). Dazu erstellen Sie ein leeres Raster und gehen alle Punkte durch. Diese werden jeweils einer bestimmten Zelle auf dem Raster zugeordnet. Wenn Sie jedem Punkt eine eigene inkrementelle z-Achsen-Koordinate geben, können Sie so später die Population bestimmen.

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.cm as cm
####read data
lats=[]
lons=[]
a = open('bath.csv', 'r')
for line in a.readlines():
    lats.append(float(line.split(',')[0]))
    lons.append(float(line.split(',')[1]))
print 'Data points: ', len(lats)
xmin=min(lons)
xmax=max(lons)
ymin=min(lats)
ymax=max(lats)
yrange=ymax-ymin
xrange=xmax-xmin
maxrange=max(xrange, yrange)
stepsize=maxrange/100
limit =int(len(lats)/20)
print 'area mapped:'
print xmin, xmax
print ymin, ymax
#define a new grid
xi = np.linspace(xmin,xmax,int(xrange/stepsize))
yi = np.linspace(ymin,ymax,int(yrange/stepsize))
zi = np.ones((len(yi), len(xi)))
print 'xi dimension ' +str(len(xi))
print 'yi dimension ' +str(len(yi))
print 'zi dimension '+ str(zi.shape)
for yindex in range(len(lats)):
#find which box it goes into
        xx=lons[yindex]-xmin
        xx=int(xx/stepsize)-1
        yy=lats[yindex]-ymin
        yy=int(yy/stepsize)-1
        if zi[yy, xx] < limit:
            zi[yy, xx]+=1
```

Weiteres Kartenlesen

Schauen Sie sich Basemap (**http://matplotlib.org/basemap**) an, mit dem das Plotten einfacher ist, wenn es um echte Karten geht. Die Flickr-API-Dokumentation (**www.flickr.com/services/api**) besitzt nicht nur sehr viele Informationen, sondern man kann auch API-Aufrufe über die Webseite ausführen. Informieren Sie sich auch über WOEIDs und über Yahoo GeoPlanet (**http://tinyurl.com/yfl7d4b**). *Matplotlib* (**http://matplotlib.sourceforge.net**) ist ein Python-Modul für das Plotten. Besuchen Sie weiterhin **http://stuvel.eu/projects/flickrapi** für das beste Flickr-Interface für Python.

```
Plot = plt.contourf(xi,yi,zi,15,cmap=plt.cm.jet)
plt.savefig('bath.svg')
plt.show()
```

Quick-Tipp

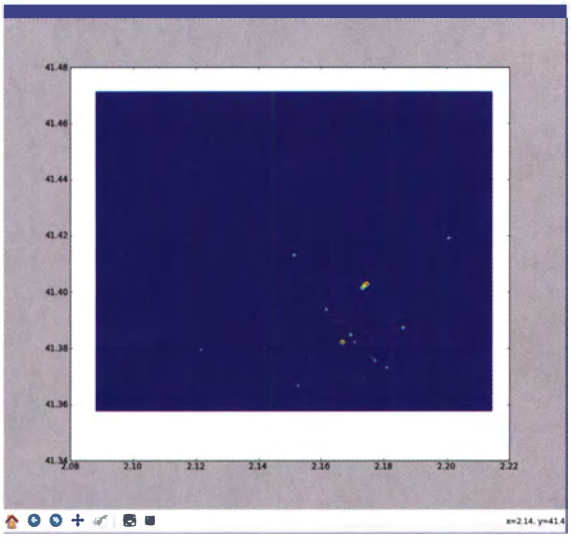
Das **plt.show()**-Kommando von **Matplotlib** öffnet ein Fenster und hält praktisch das Skript an, während ein Bild dargestellt wird. Nutzen Sie dieses Kommando am Ende einer Applikation.

Finales Tuning

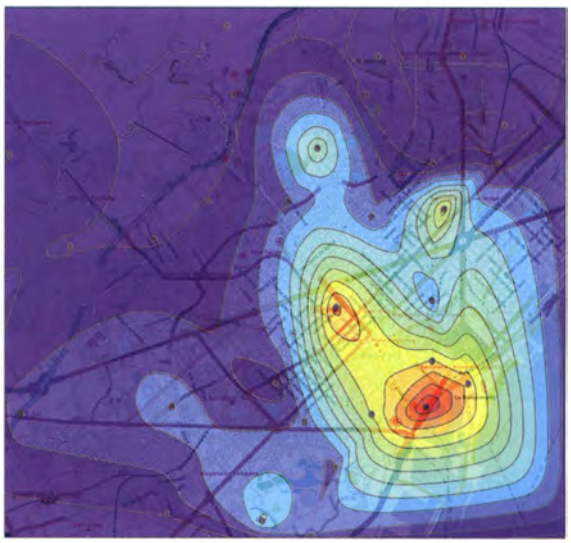
Es gibt noch Verbesserungspotenzial. Die **limit**-Variable wurde eingeführt, weil viele Bilder die gleichen Koordinaten zu besitzen scheinen. Auch wenn Sie auf einer logarithmischen Ebene plotten, die schiere Anzahl dieser Konzentrationen verwäscht die Variationen mit anderen Bereichen auf der Karte. In diesem Skript setzen wir ein Limit, welches sich aus der Gesamtzahl der Bilder in einem Bereich ergibt. Sie können etwas herumprobieren, wir haben aber meist mit 20 % die besten Ergebnisse erzielt. Die Anzahl an Rechtecken im Raster ist ebenfalls etwas, mit dem Sie experimentieren sollten:

```
stepsize=maxrange/100
```

Dieser Code gibt Ihnen 100 Intervalle über die größte Ausdehnung der Karte. Die andere Dimension wird berechnet. Wenn Ihnen 100 nicht genug ist, sollten Sie bedenken, dass ein höherer Wert auch die Ergebnisse weiter „aufkörnt". Dies erzeugt auch mehr Unruhe in dem Kartenausschnitt. Wenn die Schrittgröße zu groß gewählt wird, haben Sie fast binäre Daten auf dem Schirm. ■



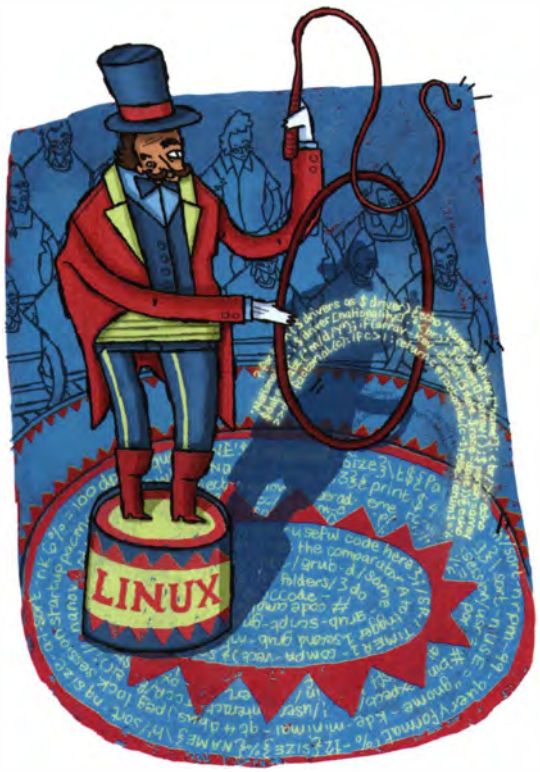
› **Hier sehen Sie Rohdaten für Barcelona. Richtig Sinn ergeben diese allerdings nur, wenn man das Ganze mit einer Karte hinterlegt.**



› **Wenn Sie die Nachbarschaftszählung verwenden, erhalten Sie zwar eine gute Heatmap, aber die Informationen sind nicht besonders detailliert.**

OAuth: Sicher

Nick Veitch zeigt Ihnen, wie Sie mithilfe des OAuth-Protokolls Ihre Login- und Passwortinformationen schützen und trotzdem identifizierungspflichtige Zusatzdienste in Anspruch nehmen können.



› **Twitter implementiert OAuth sehr sauber. Sie können Tweets also auch per Python-Skript absetzen.**

Sicherheitsaspekte sind von großer Bedeutung, wenn es um Anwendungen mit Web-Funktionalität geht. Selbst wenn Sie auf eine Website zugreifen, die keine geheimen oder finanziellen Informationen von Ihnen speichert, möchten Sie vermutlich dennoch nicht, dass Ihre Anmeldedaten von dieser Seite an Dritte weitergegeben werden, um sich nicht unnötig irgendwelchen Gefahren – von Spam bis hin zu Identitätsdiebstahl – auszusetzen.

> „Key und Secret sind zufällig generierte, temporäre Zeichenfolgen."

Das Problem ist jedoch, wenn Sie die Weitergabe dieser Daten unterbinden, können Sie viele Zusatzfunktionen von anderen Providern (Diensten) wie zum Beispiel Twitter nicht nutzen und müssen sich diese entgehen lassen – etwa Kurzlinks (URL Shortener) und Bildlinks.

Es gibt eine Lösung für dieses Problem, und das sogar schon seit Jahren, allerdings ist sie bisher nicht besonders bekannt: OAuth. Bei OAuth handelt es sich um ein Protokoll nach offenem Standard, bei dem eine Anwendung und ein Provider Informationen in Form von Keys und Tokens austauschen, welche *nicht* den Logindaten des Users entsprechen oder diese beinhalten, die jedoch trotzdem von dem jeweiligen Dienst akzeptiert werden, sofern der User sie autorisiert.

OAuth kann von Web-Anwendungen oder lokalen Einzelprogrammen eingesetzt werden. Die benutzende Anwendung wird dabei als Consumer oder Client bezeichnet. Der Consumer ist bereits bei dem adressierten Provider registriert und besitzt zwei Datenstücke, nämlich einen Key und ein Secret. Bei beiden handelt es sich um zufällig generierte, temporär gespeicherte Zeichenfolgen. Wenn die Consumer-Anwendung auf Informationen des Users zugreifen möchte, schickt sie ihren Key an den Provider und fordert von diesem ein weiteres Datenstück an, ein Token. Der User muss daraufhin dem Consumer den Zugriff erlauben, also diesen autorisieren, und sobald das geschieht, kann der Consumer sein gerade erhaltenes temporäres Token in ein Zugangs-Token umwandeln. Jede darauffolgende Anforderung seitens der Consumer-Anwendung an den Provider wird mithilfe dieses Zugangs-Tokens signiert (üblicherweise mithilfe sicherer Hash-Algorithmen) und dadurch verifiziert. Verschiedene Provider speichern, verwenden und akzeptieren diese Zugangs-Tokens auf unterschiedliche Weise, doch meistens werden die Zugangs-Tokens dauerhaft eingesetzt. Das heißt, die Zugangs-Tokens können nach ihrer Zuteilung wiederverwendet werden, bis der User die Autorisation gegebenenfalls widerruft.

Twitter und OAuth

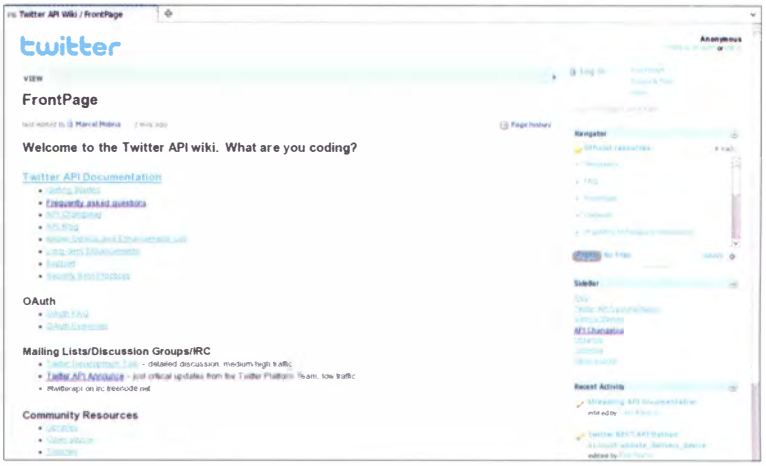
Seit etwa April 2009 unterstützt Twitter mit seiner Programmierschnittstelle (API) eine relativ robuste und geradlinige OAuth-Methode zur Autorisierung. Falls Sie selbst diese Methode anwenden möchten, müssen Sie ein Twitter-Konto besitzen und ein Set von Keys für Consumer/Clients anfordern. Das funktioniert recht einfach: Gehen Sie auf **http://twitter.com/oauth_clients** und beantworten Sie einige Fragen. Wichtig ist dabei, dass Sie

einloggen

ein Häkchen beim Punkt Athentifizierung durch Twitter setzen und den Client-Button wählen. Nachdem Sie Ihre Daten eingegeben haben, bekommen Sie Informationen zu den Anfrage-URLs angezeigt sowie die wichtigen Angaben Key und Secret. Sie benötigen diese Angaben, um Ihren Client identifizieren und autorisieren zu können, daher sollten Sie sie irgendwo abspeichern oder notieren. Sie benötigen außerdem die *Python OAuth*-Bibliothek von Leah Culver. Laden Sie die Bibliothek von der Repository oder von **http://pypi.python.org/pypi/oauth/1.0.1** herunter.

Sehen wir uns nun den ersten Schritt an, den wir ausführen müssen. Wir möchten uns mit der Twitter-API verbinden und ein Token anfordern. Öffnen Sie das Terminal, rufen Sie mit dem Befehl **python** die interaktive Python-Kommandozeile auf, und schreiben Sie:

```
>>> import httplib
>>> import oauth.oauth as oauth
>>> SERVER='twitter.com'
>>> REQUEST_TOKEN_URL='http://twitter.com/oauth/
request_token'
>>> signature_method=oauth.OAuthSignatureMethod_
HMAC_SHA1()
>>> ckey='iUMGag7VIFbQblsF1Gg'
```



› Das Wiki der Twitter-API ist eine wertvolle Quelle für Informationen und Ressourcen und unverzichtbar bei der Ermittlung der Endpunkte für die API-Methoden.

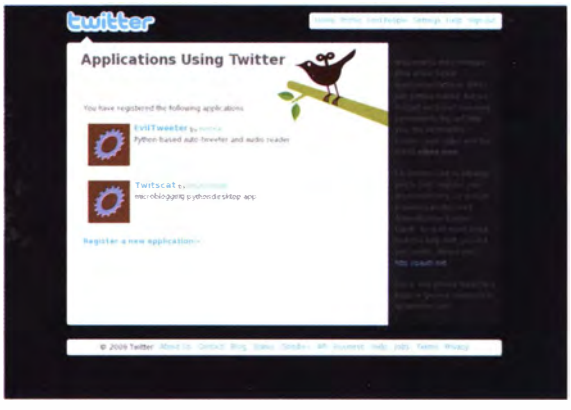
```
>>> csecret='4upg7CTuyBVkC3SIY5F5YShnf8tlxE2DtiQdRsek4'
>>> consumer = oauth.OAuthConsumer(ckey, csecret)
```

Hier haben wir zunächst *httplib* importiert, um Web-Anfragen und -Antworten zu ermöglichen, dann natürlich auch das OAuth-Modul für Python, das mit etlichen hilfreichen Klassen und Methoden ausgestattet ist. Außerdem haben wir einigen im Folgenden häufiger vorkommenden Variablen einen Wert zugewiesen, darunter natürlich auch die von Twitter erhaltenen Key und Secret sowie die URL, auf die wir zugreifen wollen.

OAuth unterstützt verschiedene Arten von Signaturen – damit ist das Hashing der Anfrage gemeint, das dazu dient, Man-in-the-middle-Angriffe abzuwehren. Da Twitter SHA-1 unterstützt, ist dies die Hashing-Methode unserer Wahl. Zuletzt haben wir noch ein Objekt für den OAuth-Consumer kreiert. Dieses benötigt die Werte für den Key und das Secret. Mit diesen Angaben können wir nun ein Anfrageobjekt erstellen:

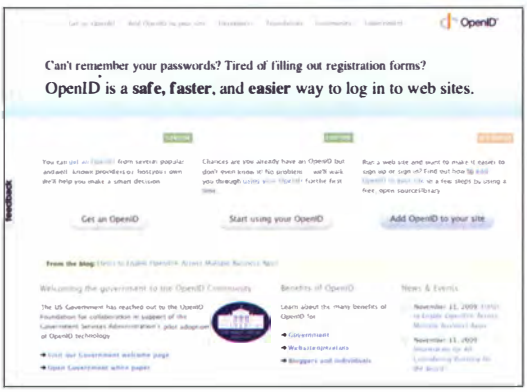
```
>>> oauth_request = oauth.OAuthRequest.from_consumer_
and_token(consumer, http_url=REQUEST_TOKEN_URL)
>>> oauth_request.sign_request(signature_method,consumer,
None)
```

»



› Wenn Sie eine Client-Anwendung registriert haben, wird diese in der Liste auf http://twitter.com/oauth_clients aufgeführt.

OpenID oder OAuth?



Da es ab und zu Verwirrung um die Begriffe OAuth und OpenID zu geben scheint, hier eine kurze Klarstellung: OpenID ist kein mit OAuth konkurrierendes System, sondern OpenID dient der Authentifizierung und OAuth der Autorisierung. Das OpenID-System ermöglicht es Anwendern, Ihre Identität zu verifizieren und so mehrere Benutzerkonten mit einem einzigen Usernamen und Loginnamen zu verwalten. Mit OAuth hingegen erlaubt man bestimmten Anwendungen, auf das eigene Benutzerkonto zuzugreifen. OpenID und OAuth schließen sich also nicht gegenseitig aus. Im Gegenteil, es sollen sogar OAuth-Erweiterungen für OpenID entwickelt werden, sodass beides im Zusammenspiel verwendet werden kann.

```
>>> oauth_request.to_header()
{'Authorization': 'OAuth realm="", oauth_nonce="30688447",
oauth_timestamp="1257173501", oauth_consumer_
key="iUMGag7VIFbQblsF1Gg", oauth_signature_
method="HMAC-SHA1", oauth_version="1.0", oauth_signatur
e="e%2B5u%2FfrenHpKmoME6r9YEtolVek%3D"'}
```

Das Anfrageobjekt hat eine Methode für seine Signatur, die die Methode, den Consumer und das Token als Parameter verwendet. Da diese Erstanfrage darin besteht, das Token anzufordern, brauchen wir das Token diesmal nicht für die Signatur zu verwenden, darum setzen wir den Wert auf **None**. Die **to_header**-Methode spuckt die Anfrage in Form eines Python-Dictionarys aus, so sind alle Elemente beisammen.

Wie Sie sehen können, ist die Signatur gehasht und eingetragen worden, und noch einige andere Werte werden generiert, am interessantesten davon wohl **nonce** und **timestamp**. Bei **nonce** handelt es sich um eine einmal zu benutzende Zahl, sie dient als eindeutige ID im Rahmen unserer Anfrage und wird vom OAuth-Modul aus Zufallszahlen generiert.

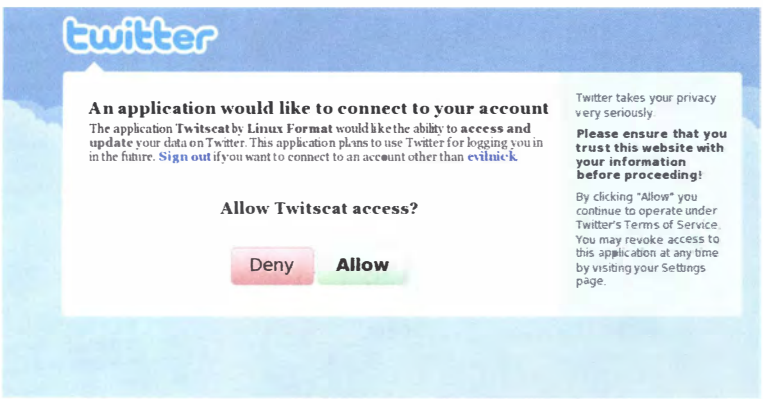
Der Zeitstempel **timestamp** dient der Sicherheit vor Ausspähangriffen, indem er eine spätere Wiederverwendung der Anfragepakete unterbindet. Ob Anfragen mit abgelaufenem Zeitstempel allerdings tatsächlich abgewiesen werden, wird vom jeweiligen Provider festgelegt.

Mit Twitter verbinden

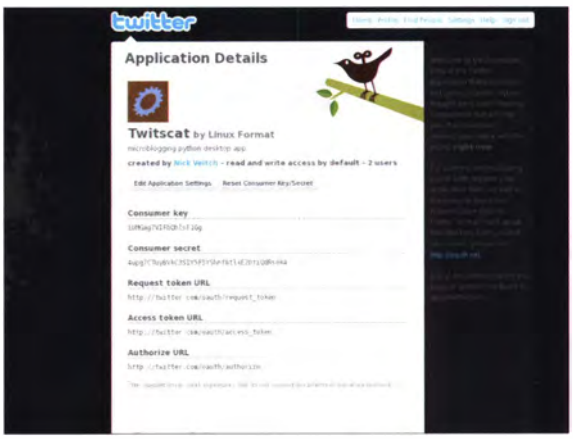
Nun da wir die Anfrage vorbereitet haben, müssen wir eine Verbindung zum Twitter-Server herstellen und sie abschicken:

```
>>> connect=httplib.HTTPConnection(SERVER)
>>> connect.request(oauth_request.http_method, REQUEST_
TOKEN_URL, headers=oauth_request.to_header())
>>> z = connect.getresponse()
>>> y =z.read()
>>> y 'oauth_token=H9Inidy04ztBBdGaCx9SYvkiMZe0TuUjiCe
i5tnM6j8&oauth_token_secret=Mw3OXlkEndaf2KeVmbLM47Cj
s7pEjShVLYrCxmfPe70&oauth_callback_confirmed=true'
>>>
```

Wir haben zuerst das Objekt **connect** erstellt und anschließend damit die Anfrage an die zuständige URL gesendet. Daraufhin benötigen wir die Antwort des **connect**-Objekts. Normalerweise würden wir damit etwas tun wollen, aber nur um zu demonstrieren, dass es funktioniert, benutzen wir die **response.read()**-Methode, um den Inhalt anzuzeigen. Wie Sie sehen können, sind **token** und **token_secret** darunter. Um die Zeichenkette später verwenden zu können, haben wir sie einer temporären Variable zugeordnet, denn die **getresponse**-Methode funktioniert wie ein Puffer (der durch das Auslesen geleert wird).



Wenn Sie um die Autorisierung eines Tokens bitten, sollte Ihr Browser Sie auf diese Twitter-Webseite leiten.



Sie haben die Möglichkeit, die Keys erneut zu generieren.

Der erste Schritt der OAuth-Prozedur ist damit abgeschlossen: Wir haben ein unspezifisches Token von Twitter zurückerhalten. Im nächsten Schritt müssen wir die Tokens aus der Antwort extrahieren und eine weitere Anfrage an Twitter stellen, diesmal geht es um das Zugangs-Token. Damit dies generiert und dem Consumer zur Verfügung gestellt werden kann, muss zuerst eine Autorisierung seitens des Users stattfinden. Hierzu schickt Twitter uns eine URL:

```
>>> token = oauth.OAuthToken.from_string(y)
>>> AUTHORIZE_URL='http://twitter.com/oauth/authorize'
>>> url = AUTHORIZE_URL + '?oauth_token=' +token.key
>>> url 'http://twitter.com/oauth/authorize?oauth_token=r9
GStmWu4BSVJtCFIt3CPhXyOClaGC2EmXinSDQE4I'
>>> import webbrowser
>>> webbrowser.open(url)
True
```

Wir können eine **oauth**-Methode anwenden, um das Token in seine Einzelelemente aufzusplitten. Was wir dabei an diesem Punkt benötigen, ist der Key, nicht das Secret. Wir fügen diesen als Variable an das Ende der Autorisierungs-URL an. Diese URL haben wir von der Twitter-Seite bekommen, über die wir unsere Anwendung registriert haben.

Um das Ganze zu vereinfachen, importieren wir das Webbrowsermodul von Python. Wenn wir die **open(url)**-Methode dieses Moduls aufrufen, öffnet sich die URL im eingestellten Standardbrowser auf dem Desktop. Indem sie sich öffnet, verarbeitet Twitter das von uns übermittelte Token und antwortet mit einem Verifizierungscode (PIN), der im

Mehr zu OAuth

Die OAuth-Protokolle sind prinzipiell überschaubar, aber es kann ein wenig Zeit erfordern, sie zu implementieren. Wenn Sie sich eingehender mit OAuth beschäftigen möchten, steht Ihnen auf der Website **oauth.net** eine umfangreiche Dokumentation nebst Code-Repository zur Verfügung. Auch ein Community-Wiki ist dort zu finden.

Browserfenster angezeigt wird. Diesen Code müssen Sie kopieren, um damit dann ein qualifiziertes Zugangs-Token anfordern zu können.

```
>>> verifier= 8650863
>>> ACCESS_URL='http://twitter.com/oauth/access_token'
>>> oauth_request = oauth.OAuthRequest.from_consumer_
and_token(consumer,token=token,verifier=verifier,http_
url=ACCESS_URL)
>>> oauth_request.sign_request(signature_
method,consumer,token)
>>> oauth_request.to_header()
{'Authorization': 'OAuth realm="", oauth_nonce="28240227",
oauth_timestamp="1257179609", oauth_signature_
method="HMAC-SHA1", oauth_consumer_
key="iUMGag7VIFbQblsF1Gg", oauth_verifier="8650863",
oauth_version="1.0", oauth_token="DqW8aSn6dXoNS5SwPcX
bs6hl4KR5vPIXFZ5eFNBdQ8", oauth_signature="hatOW%2Bl4
oSp2f8JTZ5oaWqrsFDs%3D"'}
>>> connect.request(oauth_request.http_method, ACCESS_
URL,headers=oauth_request.to_header())
>>> z =connect.getresponse()
>>> y=z.read()
>>> y
'oauth_token=16151271-2OGXeCrHlzmSyCrclWd1MHMbTFLgv
A4dS2it6m0Nx&oauth_token_secret=BDBU01aLvjIxECtxHLziz
57NM7VWYK1USOf93KsW4U&user_id=16151271&screen_
name=evilnick'
>>>
```

Als Erstes müssen wir den Verifizierungscode als Variable eingeben und die URL für die Annahme des Tokens aufsetzen. Erneut haben wir ein **oauth_request**-Objekt erstellt, doch diesmal besitzen wir das Token und den Verifizierungscode, um Einlass zu finden. Außerdem müssen wir, wenn wir dieses Mal die Anfrage signieren, das erhaltene Token anwenden. Wir können die Header anzeigen lassen, um zu sehen, wie die Anfrage aussehen wird.

Wir senden die Anfrage wiederum mithilfe von *httplib* und filtern die Antwort in **y** als temporäre Variable hinein. Wie Sie der Antwort entnehmen können – ebenso wie **token** und **token_secret** –, sind die User-ID und der Username für den Account zurückgekommen. Wir können das Token mittels der **oauth_token**-Methode extrahieren, aber die OAuth-Bibliothek weiß erst einmal nichts von der Existenz von Twitter an sich. Darum müssen wir uns die anderen Informationen, die wir haben möchten, selbsttätig besorgen.

Jetzt haben wir das Token samt Autorisierung bekommen, aber funktioniert es auch? Zum Glück gibt es einen Endpunkt der Twitter-API für diese Aufgabe, zu finden unter **twitter.com/account/verify_credentials**. Die Endpunkte liegen in den Formaten JSON (JavaScript Object Notation) und XML vor. Wir benutzen hier das serialisierte JSON-Format.

Da das neue Token sich vom vorherigen unterscheidet, müssen wir es erneut einlesen. Bei Twitter (wenn auch nicht zwangsläufig bei anderen Diensten) wird das Zugangs-Token dauerhaft verwendet, darum sollten Sie es speichern.

Diesmal konstruieren wir eine Anfrage basierend auf der URL, auf die wir zugreifen möchten, und signieren sie wie gehabt. Diese URL benötigt lediglich die **get**-Methode, da es sich nur um einen Lesezugriff handelt. Wir verbinden uns erneut mit Twitter und empfangen die Antwort. Das serialisierte JSON-Objekt, das wir zurückbekommen, kann entschlüsselt werden und versorgt uns mit diversen Informationen über den betreffenden Account.

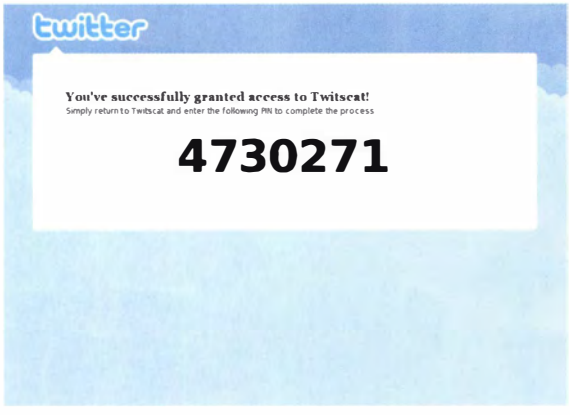
Der Praxistest

Um zu sehen, ob das Ganze auch in der Praxis funktioniert, führen wir zum Schluss noch einen Test durch. Wir wollen versuchen, auf der Twitter-Plattform einen neuen Tweet abzusetzen. Hierzu müssen wir die erforderlichen Parameter sowie die Ziel-URL kennen. Die entsprechenden Informationen findet man in der Dokumentation der Twitter-API.

> „Das Modell kann auf jede API-Funktionalität angewandt werden."

Um die Parameter einfügen zu können, erstellen wir ein neues Dictionary-Objekt und fügen es unserer Anfrage hinzu. Dies ist unbedingt erforderlich, denn die Parameter müssen in den String integriert werden, der dann gehasht wird, um die Signatur zu erzeugen – andernfalls funktioniert es nicht. Da wir in diesem Fall Daten schreiben, wenden wir eine **post**-Operation an, die in die Anfrage aufgenommen werden muss. Die Anfrage versenden wir wieder mithilfe von *httplib*, allerdings wegen der **post**-Operation mit einer etwas anderen Syntax: Wir brauchen diemal eine serialisierte Version von **oauth_request** statt eines Headers. Im *Python OAuth*-Modul gibt es eine **to_postdata()**-Methode für **request**-Objekte, die die Serialisierung bewerkstelligt. OAuth-Anfragen können auf zweierlei Arten durchgeführt werden: Mithilfe von **get** beim Lesezugriff (wobei die Anfrage signiert und die Tokens als Header angefügt werden) oder mittels **post**, wobei die OAuth-Details mit den Daten serialisiert werden und als Parameter an die URL geschickt werden. Als Antwort wird Twitter das vollständige serialisierte Statusobjekt zurückschicken – dieses enthält alle möglichen Daten betreffend den User sowie den soeben gepostete Tweet.

Das hier erklärte Modell kann prinzipiell auf jede andere API-Funktionalität angewandt werden, sei es bei Twitter oder bei einem anderen Provider – wobei jedoch nicht jeder Dienst OAuth genauso sauber implementiert. ■



› Sofern der notwendige Autorisierungsprozess die neueste Spezifizierung von OAuth unterstützt, müssen Sie einen solchen Verifizierungscode (PIN) eingeben.

Quick-Tipp

Falls Sie im Laufe Ihrer Versuche mit OAuth unkommentierte Fehlermeldungen erhalten sollten, dann wiederholen Sie den gleichen Schritt einfach noch mal. Manchmal gehen Anforderungen unter. Denken Sie aber daran, dass Sie jedes Anfragepaket nur einmal benutzen können. Sie müssen dann also ein neues generieren.

SFT MIT FILM-DVD | COMPUTER | HANDY | TV | FOTO | AUDIO | INTERNET | WWW.SPIELEFILMETECHNIK.DE

2 TOP-MOVIES AUF HEFT-DVD

SFT

SPIELE | FILME | TECHNIK

25 JAHRE COMPUTEC

Highend-Schnäppchen: Google Nexus 5

Brandneues Premium-Handy mit Android 4.4 zum Budget-Preis

Seite 32

1. FILM uncut

„Atemberaubende Bilder mit hypnotischer Kraft" (BR Kino Kino)

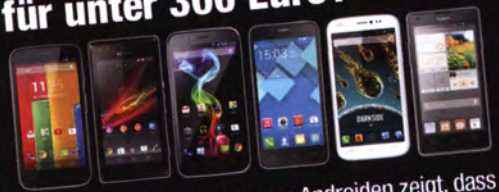
2. FILM

„Magnolia und L.A.Crash sind Vorbilder" (Video.de) „Bewegendes Drama" (Movieman.de)

Besserer Sound für Ihren TV!

IM TEST: 13 Soundsysteme unter 500 Euro

Top-Smartphones für unter 300 Euro?



TEST: Unser Check von neun Androiden zeigt, dass gut nicht unbedingt auch teuer sein muss. Seite 36

70 Zoll: Riesig fernsehen!

TEST: Wie gut ist das Bild vom Sharp LC-70LE857E? Seite 50

TEST: Hightech für Kopf und Arm

Wir zeigen, was Samsungs smarte Uhr Galaxy Gear und Sonys 3D-Kino-Helm HMZ-T3W draufhaben. Seite 42 Seite 52

TEST: Neue Tablet-Computer

Drei Tablets mit Top-Leistung: Samsung Galaxy Note 10.1, LG G Pad 8.3 und Apples iPad Mini mit Retina Display. Seite 46

FSK ab 16 freigegeben

110. AUSGABE
01/14 | Januar
€ 4,99
Erhältlich auch ohne DVD für € 3,50
Deutschland € 4,99; Österreich € 5,70; Schweiz sfr 8,80; Holland, Belgien, Luxemburg € 5,90; Frankreich, Italien, Spanien, Portugal, Griechenland € 6,60

DAS TEST-MAGAZIN
FÜR DIGITALE UNTERHALTUNG

DAS BESTE AUS ALLEN TECHNIK-WELTEN:

Flat-TVs | Smartphones | Tablets | Notebooks | Digitalkameras | Video | HiFi

plus: **DIE COOLSTEN SPIELE** und **ALLE FILM-BLOCKBUSTER**

2 TOP-MOVIES AUF HEFT-DVD

Auch als Magazin-Variante ohne DVD für € 3,50

25 JAHRE computec MEDIA

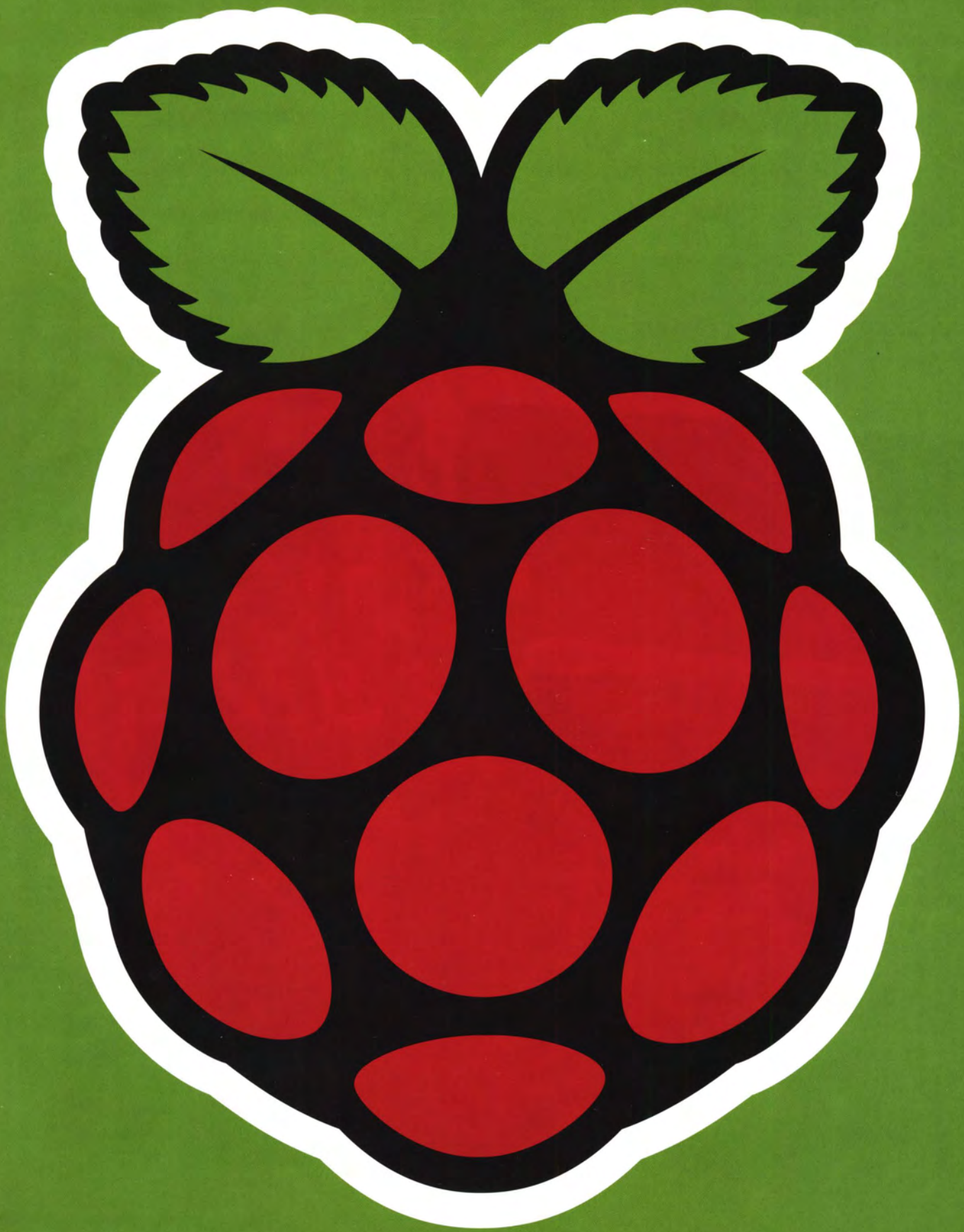
QR-Code scannen und hinsurfen!

WWW.SPIELEFILMETECHNIK.DE

Neue Ausgabe jetzt am Kiosk erhältlich oder einfach online bestellen unter: shop.spielefilmetechnik.de

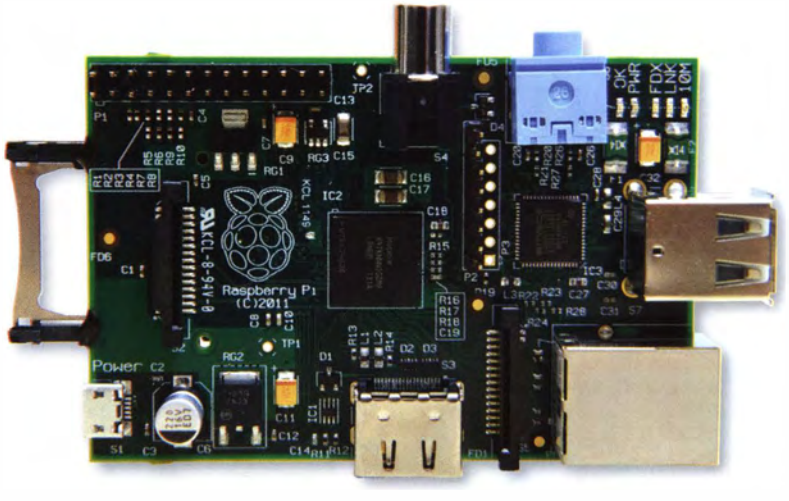
Auch erhältlich als ePaper bei:

computec KIOSK | iKiosk | Kiosk online | PresseKatalog Online-Kiosk



Linux

Das Standard-Betriebssystem des Raspberry Pi heißt Raspbian, und Raspbian ist eine der vielen Varianten von Linux. Allerdings läuft Linux bei Weitem nicht nur auf dem Raspberry Pi, sondern steckt auch in allen möglichen anderen Geräten vom Smartphone bis zum Superrechner. In diesem Abschnitt bekommen Sie eine umfassende Einführung in Linux sowie einen achtteiligen Lehrgang vom Profi.



Was ist Linux?

Neil Bothwick untersucht das weltbeste Betriebssystem und legt offen, was es antreibt.

Linux – was genau ist damit eigentlich gemeint? Die Bedeutung variiert, je nachdem, wen man fragt: Der Purist antwortet, dass der Begriff nur für den Kernel steht. Für den GNU-Verfechter ist das Wort ein Teil von GNU/Linux. Der Linux-Anfänger denkt dagegen, es wäre nur ein anderer Name für Ubuntu. In Wahrheit ist Linux aber jeder dieser Punkte, nur die Sichtweise wechselt. Genau genommen bezieht sich der Begriff Linux nur auf den Kernel des Betriebssystems, während GNU/Linux das gesamte Betriebssystem bezeichnet, das den Linux-Kernel zusammen mit den GNU-Werkzeugen enthält – das Eine wäre nutzlos ohne das Andere (oder ohne eine der Alternativen).

Wenn wir dem Betriebssystem noch eine Sammlung von Anwendungsprogrammen beilegen, zusammen mit einigen Tools, um das Ganze zu verwalten, haben wir eine Distribution. So wie beispielsweise Ubuntu.

Hier erklären wir, wie Linux sich aus Einzelteilen zusammensetzt und was diese Komponenten genau machen.

Was ist ein Betriebssystem? Und was eine Distribution?

Ein Betriebssystem kann als eine Software definiert werden, die für andere Software nötig ist, um auf der Hardware laufen zu können. Ein Betriebssystem besteht aus mehreren Schichten. Ganz unten haben wir den Kernel, der mithilfe von Treibern direkt mit der Hardware interagiert und es so der anderen Software ermöglicht, die Hardware zu verwenden. Andere Schichten setzen auf dieser Basis auf, beispielsweise jene, die Dinge wie Eingabegeräte, Netzwerk, Audio- und Video-Ausgabe verwalten.

Normalerweise müssen Sie nichts über diese Elemente wissen. Allerdings kann es sehr hilfreich sein, wenn mal etwas nicht so funktioniert, wie es sollte. Aber selbst dann ist Wissen über diese Komponenten keineswegs Voraussetzung. Besonders dann, wenn Sie jemanden auftreiben können, der Ihren Computer für Sie repariert. Da Sie aber gerade dieses Bookazine lesen, stehen die Chancen recht gut, dass Sie sich für die Vorgänge „da unten“ interessieren. Deshalb versuchen wir, Ihnen einen Überblick zu geben, was wo abläuft und wie es das macht.

Eine Linux-Distribution ist genau das, was der englische Begriff „to distribute“ bedeutet: Das Verteilen eines linuxbasierten Betriebssystems zusammen mit weiteren Softwarebeigaben. Zu Beginn waren das nur die Dateien, die das Betriebssystem benötigt, zusammen mit einer Methode, das Ganze auf einem Computer zu installieren. Mit der Zeit wurden die Distributionen um Paket-Manager, Update-Tools, Oberflächen zur Konfiguration und einem Haufen anderer Annehmlichkeiten erweitert. Unter der der benutzerfreundlichen (oder auch nicht, falls Sie Gentoo-Nutzer sind) Haube sind aber alle Distributionen nach wie vor ein normales Linux.

Der Kernel

Das Nervenzentrum im Herzen Ihres Linux-Betriebssystems.

Der Kernel ist das schlagende Herz des Systems, aber was bedeutet das genau? Er kommuniziert im Namen der gesamten auf dem Computer installierten Software mit dem Prozessor, dem Speicher und all den anderen Geräten. In dieser Rolle ist der Kernel die am tiefsten gelegene und gleichzeitig wichtigste Software-Schicht. Falls der Kernel ein Problem hat, hat das Problem auch die gesamte Software auf dem Rechner.

Der Linux-Kernel ist monolithisch aufgebaut. Das bedeutet, dass alle Kerndienste des Betriebssystems im Kernel laufen. Die gegenteilige Herangehensweise wäre ein Micro-Kernel, wo ein Großteil der Arbeit auf externe Dienste ausgelagert wird und der Kernel kaum Weitergehendes macht, als die Aufgaben zu koordinieren.

Während ein reinrassiger monolithischer Kernel in den Anfangstagen der Betriebssysteme gut funktionierte, ist das Prinzip in Reinform heute nicht mehr praktikabel: Aufgrund der riesigen Hardwarevielfalt wäre ein Kernel, der alle ihre Kombinationen komplett abdeckt, viel zu groß. Deshalb ist der Linux-Kernel von heute modular. Die Kernfunktionen befinden sich in der Kernel-Datei (zu finden in **/boot** als **vmlinuz-version**) während sich die optionalen Treiber als separate Module im Verzeichnis **/lib/modules** befinden (die **.ko**-Dateien in diesem Verzeichnis).

Ziehen wir als Beispiel für den Speicherbedarf Ubuntu 12.10 heran: Während der Kernel nur 5 MB groß ist, belegen die zusätzlichen 3.700 Moduldateien 100 MB. Da auf jedem Computer nur ein Bruchteil dieser Dateien benötigt wird, wäre es unsinnig, sie alle mit dem Kernel zu laden. Stattdessen ermittelt der Kernel, welche von diesen Modulen er braucht. Diese werden geladen und sind im Hauptspeicher Teil des Kernels. Im Betrieb ist der Kernel also immer noch monolithisch, auch wenn er über Tausende Dateien verteilt ist. So ist es dem System möglich, auf eine sich verändernde Hardware zu reagieren. Wenn Sie beispielsweise einen USB-Stick einstöpseln, lädt das **usb-storage**-Modul – zusammen mit dem Dateisystem-Modul, das Sie benötigen, um den Stick mounten zu können. Wenn Sie einen 3G-Dongle einstecken, werden die Treiber für das serielle Modem geladen. Das ist der Grund, warum Sie so selten neue Treiber installieren müssen, wenn Sie neue Hardware mit dem Computer verbinden: Die meisten Treiber sind schon vorhanden und warten nur darauf, dass Sie ein passendes Gerät anstecken.

Computer, die auf einer spezifischen, sich nicht verändernden Hardware laufen – wie etwa Server – nutzen meist einen Kernel, der mit allen nötigen Treibern kompiliert wurde. So kann das Nachladen von Modulen deaktiviert werden, da das eine kleine Verbesserung bei der Sicherheit bringt.

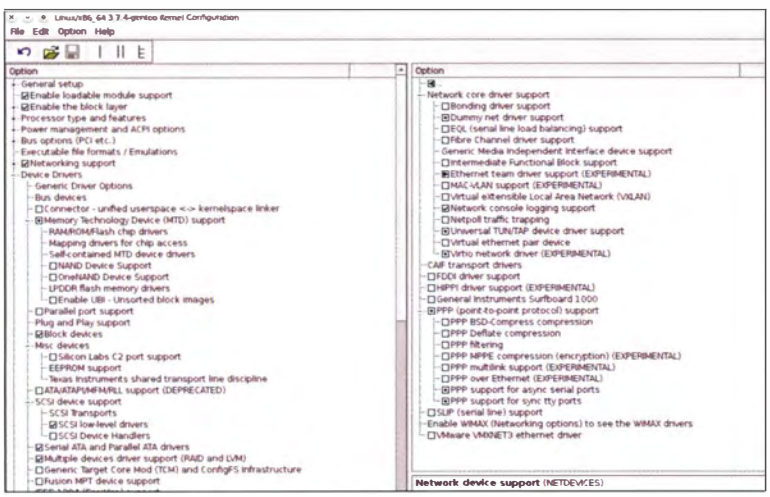
Falls Sie Ihren eigenen Kernel kompilieren, gilt die Faustregel, dass Sie alle Treiber für jene Hardware integrieren sollten, die ständig im Gebrauch sein wird, wie etwa für Netzwerke und Dateisysteme. Für alles andere greifen Sie besser auf Module zurück.

> „Das System unterstützt extrem viel Hardware, ohne Treiber herunterladen zu müssen."

Und noch mehr Module

Obwohl Linux eine extrem gute Hardwareunterstützung bietet, fehlt mitunter ein passendes Kernel-Modul für ein Gerät, insbesondere wenn der Code sehr neu ist oder aus lizenzrechtlichen Gründen. Die darum fehlenden Treiber werden meist als Drittanbieter-Module (bei Ubuntu als „restricted drivers") bezeichnet. Falls diese von Ihrer Distribution unterstützt werden, lassen sie sich oft per Paketmanager installieren. Andernfalls müssen sie ausgehend vom Quellcode kompiliert werden, und zwar jedes Mal, wenn Sie Ihren Kernel aktualisieren, denn die Treiber sind hierbei an den Kernel gebunden, für den sie kompiliert wurden. Um den Aufwand zu verringern, gibt es automatisierte Lösungen wie DKMS (Dynamic Kernel Module Support), das bei einer Neuinstallation des Kernels alle Drittanbieter-Module neu kompiliert. Damit wird das Aktualisieren des Kernels fast genauso einfach wie ein Update eines Anwendungsprogramms.

Im Umgang mit Kernels werden Sie oft über Begriffe wie „Kernel-Space" und „User-Space" stolpern. Beim Kernel-Space handelt es sich um einen für den Kernel reservierten Speicherbereich. Auf diesen haben nur der Kernel und seine Module Zugriff, nicht aber Anwendungsprogramme. Damit soll verhindert werden, dass Amok laufende Programme die Arbeit des Kernels stören. Dem gegenüber steht der User-Space, der von jedem Programm angesprochen werden kann, sofern es die entsprechenden Berechtigungen dazu hat. Diese Aufteilung trägt viel zur Stabilität von Linux bei, da kein Programm die Arbeit des Kernels direkt beeinflussen kann, selbst wenn es mit Root-Privilegien läuft.



› Die Anzahl der Optionen, die Ihnen beim Eigenbau eines Kernels zur Verfügung stehen, ist überwältigend. Die meisten drehen sich dabei um den Support von Hardware. Sind Sie nicht froh, dass uns die Distributoren diese Arbeit abnehmen?

Die Startreihenfolge

Das mysteriöse Leuchten und Piepen während des Computerstarts.

Die meisten Distributionen springen beim Starten direkt zu einem Splash-Screen, sodass wir nicht erkennen können, was im Hintergrund abläuft. Dabei passiert schon viel, bevor der Splash-Screen überhaupt auftaucht und die Vorgänge versteckt. Zuerst startet das BIOS in der Hardware des Motherboards. Es sieht sich nach einem bootbaren Gerät um, von dem es Code laden kann.

Dieser befindet sich im Falle eines Festplattenlaufwerks mit klassischem DOS-Partitionssystem im nur 512 Byte großen Master Boot Record (MBR). Davon sind 64 Bytes zur Speicherung der Partitionstabelle vorgesehen (diese kleine Datenmenge ist der Grund, warum nur maximal vier primäre Partitionen möglich sind). Die restlichen 446 Byte beinhalten den Code für den Bootloader, im Falle von Linux meist *GRUB*. 446 Bytes erlauben klarerweise keinen überbordenden Funktionsumfang, weshalb der Code in diesen Speicher nichts weiter tut, als den Bootloader von irgendwo anders auf der Festplatte zu laden. Von wo genau, wird festgelegt, wenn der MBR auf die Festplatte geschrieben wird.

> „Der Bootloader entnimmt der Konfigurationsdatei den Speicherort des Kernels."

Der Bootloader liest anschließend seine Konfigurationsdatei ein, im Falle von *GRUB* **/boot/grub2/grub.cfg**. In dieser ist eine Liste von Boot-Optionen gespeichert. Diese wird je nach Konfiguration entweder dem Nutzer angezeigt oder startet sofort den als Standard markierten Eintrag.

Zu diesem Zeitpunkt ist Linux noch in keiner Weise in den Betrieb des Rechners involviert, bis jetzt wurde nur Code des Bootloaders ausgeführt. Aus der vorher genannten Konfigurationsdatei entnimmt der Bootloader nun die Informationen zu den Speicherorten des Linux-Kernels, jeder **initramfs**-Datei, die er brauchen könnte, sowie auch der Root-Partition. Außerdem informiert er sich, ob das Ganze hinter einem Splash-Screen versteckt werden soll. Falls Sie sehen möchten, was von jetzt an passiert, können Sie den Splash-Screen auf den meisten Distributionen mit der E-Taste zum Bearbeiten des Eintrags im *GRUB*-Menü entfernen, indem Sie alle Quiet- und Splash-Parameter löschen. Ein Druck auf F10 fährt anschließend mit dem Bootvorgang fort.

› Das Entfernen des Splash-Screens zeigt die Vorgänge beim Bootvorgang des Computers in Gänze zusammen mit dem Status der startenden Dienste.

Wozu eine RAM-Disk?

Die meisten Distributionen benutzen eine **initramfs**-Datei. Der Hauptgrund dafür ist, dass bestimmte Treiber zusammen mit dem Kernel geladen werden müssen. Speziell geht es dabei um jene Module, die es dem Kernel überhaupt ermöglichen, irgendetwas auf der Festplatte zu lesen (beispielsweise für SATA-Controller und Dateisysteme). Würde man alle möglichen Optionen in den Kernel integrieren, wäre dieser für eine gewöhnliche Distribution unhandlich groß. Deshalb ist alles in Form von Modulen hinterlegt. Und jene, die zum Booten benötigt werden, sind in der **initramfs** integriert. Es ist eine Art von RAM-Disk, die vom Bootloader (mithilfe der BIOS-Funktionen zum Lesen der Festplatte) geladen wird und alle Dateien beinhaltet, die zum Mounten der Root-Partition notwendig sind. Die Geräteerkennung des Kernels entscheidet dabei, welche Dateien notwendig sind. Anschließend wird die Kontrolle an die richtige Festplatte übergeben. Die RAM-Disk **initramfs** lädt auch potentielle Splash-Screens, sodass sie am Beginn des Startvorganges richtig dargestellt werden.

Sobald die Root-Partition gemountet ist, beginnt erst die richtige Start-Sequenz, entweder direkt oder über das **initramfs**. Normalerweise beinhaltet das Ausführen von **/sbin/init**, worüber alles andere – wie in **/etc/inittab** festgelegt – gestartet wird. Auch alle Start-Meldungen der Dienste werden von Ersterem kontrolliert. Diese Ausgaben ermöglichen es Ihnen auch, zu erkennen, wo der Startvorgang hängt, falls er einmal scheitern oder unverhältnismäßig viel Zeit in Anspruch nehmen sollte.

Neuere Optionen

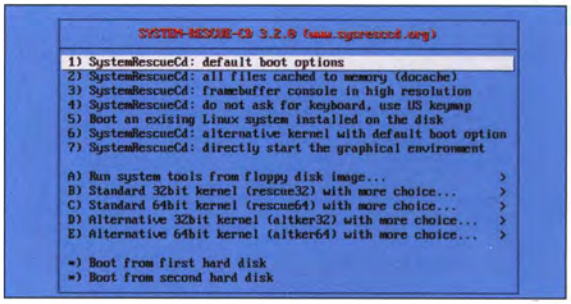
Die Zeit schreitet voran, und so werden alle diese Komponenten immer weiter modernisiert. Auf aktueller Hardware wurde das BIOS durch UEFI ersetzt. Sobald aber der Bootloader

Das GNU in GNU/Linux

Das GNU-Projekt (Akronym für GNU's not Unix) ist schon einige Jahre älter als Linux. Im Rahmen dieses Projekts wurden die meisten der grundlegenden Werkzeuge und Programme geschaffen, die für einen Computer der frühen 1980er Jahre benötigt wurden: Compiler, Text-Editoren, Kommandos zum Bearbeiten von Dateien und Verzeichnissen und vieles mehr. Doch leider fehlte dem Projekt ein benutzbarer Kernel. Als Linus Torvalds 1992 an seinem kleinen Projekt herumzubasteln begann, hatte er einen Kernel, allerdings fehlten ihm die Werkzeuge, um ihn laufen zu lassen. GNU und Linux wurden folgerichtig zusammengefügt, und GNU/Linux war geboren – ein Betriebssystem, das den Linux-Kernel und die GNU-Werkzeugsammlung verwendet. Es sind nicht nur die Programme in **/bin** und **/usr/bin**, die von GNU stammen: **glibc** stammt ebenfalls vom GNU-Projekt und ist die C-Kernbibliothek in Linux. Jedes Mal, wenn Sie etwas an ihrem Computer machen, immer wenn Sie ein Icon anklicken oder ein Kommando eintippen, hilft GNU-Software an irgendeiner Stelle der Befehlskette. Kein Wunder also, dass es die GNU-Verfechter immer auf die Palme treibt, wenn wir das Betriebssystem Linux und nicht GNU/Linux nennen. Es aber ist wichtig festzuhalten, dass niemand die Bedeutung des Anteils von GNU an GNU/Linux bestreitet! Dass wir das Betriebssystem Linux nennen, hat viel mehr mit Bequemlichkeit und Lesefreundlichkeit zu tun: Der volle Name ist einfach zu sperrig.

installiert wurde, werden Sie keine Unterschiede zwischen den beiden Systemen bemerken. Es gibt auch Bestrebungen, das klassische SysVinit-System zu ersetzen, woran schon viele Jahre gearbeitet wird. So hat Ubuntu beispielswiese *Upstart* eingeführt, während Fedora und Red Hat zu *systemd* greifen.

Obwohl sie alle die gleichen grundlegenden Aufgaben erfüllen – jene Dienste zu starten, die für den Betrieb des Betriebssystems notwendig sind – unterscheiden sie sich stark in ihrer Vorgehensweise. Der größte Unterschied ist, dass SysVinit sequenziell vorgeht. Ein Dienst muss starten, bevor der nächste auf der Liste angestoßen wird. Ist ein langsam startender Dienst dabei, hält dieser alle anderen auf. *Upstart* und *systemd* starten die Dienste dagegen parallel und vermeiden so diesen Flaschenhals. Natürlich gibt es jene, die behaupten, dass Linux so stabil ist, dass die Startzeiten nicht ins Gewicht fallen, weil man auf Funktionen wie den Ruhezustand zurückgreifen kann. Wenn Sie den Ruhezustand statt Herunterfahren nutzen, wird ein Reboot in der Tat zu einem sehr seltenen Ereignis.



› **GRUB ist der populärste Bootloader. Bei Live-CDs und USB-Sticks ist die Verwendung von *isolinux* oder *syslinux* aber gängiger.**

Bibliotheken

Warum sich Programme Funktionen teilen.

Linux verwendet Bibliotheken, um Programmcode zwischen verschiedenen Programmen zu teilen. Wenn ein Programm namens foo Funktionen nutzt, die auch irgendwoanders nützlich sein können, werden diese in libfoo abgelegt. Falls ein anderes Programm, hier „bar" genannt, die gleichen Funktionen nutzen will, muss es nur auf libfoo verweisen, anstatt das Rad neu zu erfinden.

Das bedeutet, dass sich im Idealfall nur eine einzige Ausgabe eines bestimmten Codes auf Ihrem Rechner befindet. Falls in diesem ein Fehler auftaucht, reicht das Ausbessern in diesem in einer Bibliothek enthaltenen Code-Abschnitts, um den Fehler in allen darauf verweisenden Programmen zu korrigieren. Das führt auch das Konzept der Abhängigkeiten ein: In unserem Fall sind sowohl **foo** als auch **bar** von **libfoo** abhängig und ohne diese Bibliothek nutzlos. Genau diese Abhängigkeiten sind auch für das Phänomen der „Hölle der Abhängigkeiten" verantwortlich, wo die Installation eines Programmes durch nicht erfüllte Abhängigkeiten verhindert wird und beim Nachinstallieren dieser Abhängigkeiten sich sofort neue auftun. Heute ist das aber meist nur noch eine unangenehme Erinnerung, weil Distributions-Repositorys immer umfangreicher und Paket-Manager immer zuverlässiger werden.

Wenn Sie mit dem Paket-Manager Ihrer Distribution arbeiten, wird sich dieser um alle Abhängigkeiten kümmern, ohne dass Sie einen Gedanken daran verschwenden müssen. Versuchen Sie mal, **irgendwaszufälliges.deb** oder **irgendwaszufälliges.rpm** von **www. irgendwaszufälliges.com** zu installieren, und Sie werden früher oder später feststellen, warum es besser ist, so etwas in Zukunft dem Paket-Manager zu überlassen. Eine mögliche Lösung für das Ganze ist es, die Programme statisch zu kompilieren. Das bedeutet, dass beim Kompilieren nicht auf den Code in **libfoo** verwiesen und während der Laufzeit des abhängigen Programmes dynamisch darauf zugegriffen wird, sondern der Compiler den Code von **libfoo** direkt in die abhängigen Programme integriert. Im Falle von **foo** und **bar** wären nach einem statischen Kompilierungsvorgang zwei von **libfoo** unabhängige Programme das Ergebnis – mit dem Nachteil, dass wenn in **libfoo** ein Fehler gefunden würde, beide Programme neu kompiliert und für Ihre Distribution neu bereitgestellt werden müssten. Generell ist es die bevorzugte Methode, dynamisch auf eine Bibliothek zu verlinken, sofern nicht für ein Embedded-System programmiert wird. Aber es gibt eine Situation, wo statisch verlinkte Programme nützlich sind: im **initramfs**, das beim Systemstart geladen wird. So vermeiden Sie es, Bibliotheken in das RAM-Disk-Abbild integrieren müssen. Falls es Sie interessiert, welche Programme welche Abhängigkeiten aufweisen, erfahren sie das mit dem **ldd**-Kommando.

`ldd /usr/bin/irgendeinprogramm`

zeigt alle von einem Programm benötigten Bibliotheken, sowie auch deren Abhängigkeiten an. Und fast immer endet die Aufzählung bei **libc**, dem Großvater aller Linux-Bibliotheken.

```
[nelz@hactar ~ 0]% ldd /usr/bin/k3b
        linux-vdso.so.1 (0x00007fffe61ff000)
        libk3bdevice.so.6 => /usr/lib64/libk3bdevice.so.6 (0x00007ffd9cbec000)
        libk3blib.so.6 => /usr/lib64/libk3blib.so.6 (0x00007ffd9c811000)
        libkcddb.so.4 => /usr/lib64/libkcddb.so.4 (0x00007ffd9c5cc000)
        libkfile.so.4 => /usr/lib64/libkfile.so.4 (0x00007ffd9c31b000)
        libkio.so.5 => /usr/lib64/libkio.so.5 (0x00007ffd9be4b000)
        libknotifyconfig.so.4 => /usr/lib64/libknotifyconfig.so.4 (0x00007ffd9bc370
        libkde3support.so.4 => /usr/lib64/libkde3support.so.4 (0x00007ffd9b932000)
        libsolid.so.4 => /usr/lib64/libsolid.so.4 (0x00007ffd9b636000)
        libQtWebKit.so.4 => /usr/lib64/qt4/libQtWebKit.so.4 (0x00007ffd99b7c000)
        libkcmutils.so.4 => /usr/lib64/libkcmutils.so.4 (0x00007ffd99938000)
        libQt3Support.so.4 => /usr/lib64/qt4/libQt3Support.so.4 (0x00007ffd99455000
        libQtXml.so.4 => /usr/lib64/qt4/libQtXml.so.4 (0x00007ffd99210000)
        libkdeui.so.5 => /usr/lib64/libkdeui.so.5 (0x00007ffd98b73000)
        libkdecore.so.5 => /usr/lib64/libkdecore.so.5 (0x00007ffd98682000)
        libQtCore.so.4 => /usr/lib64/qt4/libQtCore.so.4 (0x00007ffd981a4000)
        libQtGui.so.4 => /usr/lib64/qt4/libQtGui.so.4 (0x00007ffd974f3000)
        libstdc++.so.6 => /usr/lib/gcc/x86_64-pc-linux-gnu/4.6.3/libstdc++.so.6 (0x
        libc.so.6 => /lib64/libc.so.6 (0x00007ffd96e45000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007ffd96c28000)
```

› **Bibliotheken ermöglichen ein effizienteres System, indem sie Code für andere Anwendungen zugänglich machen. Hier sehen Sie nur einige von den Bibliotheken, auf die das Brennprogramm *K3b* verlinkt.**

Paket-Manager

Die große Flexibilität von Linux bedeutet, dass die meisten Elemente auch angepasst werden können. Standardanwendungen, Desktops und sogar Kernels können ausgetauscht werden, sodass es am besten ist, eine Distribution wie Fedora oder Ubuntu als Ausgangspunkt für eigene Anpassungen zu betrachten.

Die einzige Komponente, die nicht so ohne Weiteres ausgetauscht werden kann, ist der Paket-Manager, weshalb der Distributionstausch auch die einzige Möglichkeit ist, einen anderen Paketmanager auszuprobieren. Wenn Sie Yast von SUSE mit *Synaptic* von Debian vergleichen, werden Sie feststellen, wie fundamental sich so ein grundlegendes Tool auf den Umgang mit einer Distribution auswirken kann.

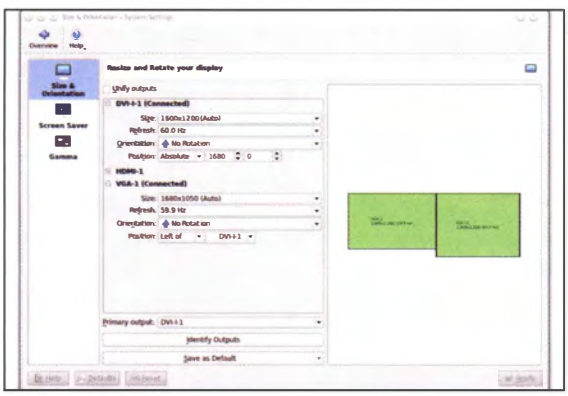
Grafik

Warum Ihr Linux-Betriebssystem so gut aussieht.

Das *X Window System* ist die Standardbasis, um unter Linux ein grafisches Benutzerinterface zu ermöglichen. Während Gnome und KDE die Benutzerschnittstelle zur Verfügung stellen, ist es *X*, über welches die komplette Darstellung realisiert wird. Eine lange Zeit war das System eine Ansammlung von kryptischen Einstelllungen und benötigte eine ellenlange Konfigurationsdatei, in der Pixeltaktraten und Synchronisationsfrequenzen festgelegt wurden.

Heute braucht man dagegen kaum noch Konfigurationsdateien. Das verdanken wir vor allem einer stark verbesserten Hardwareerkennung, sodass Systeme mit einer einzelnen Grafikkarte und einem Monitor ohne Konfiguration sofort funktionieren. Für die 3D-Beschleunigung könnte ein zusätzlicher Treiber notwendig sein, beispielsweise wenn Sie eine Grafikkarte von Nvidia nutzen. Andernfalls starten Sie Ihren PC und können sofort mit der Arbeit loslegen. *X* nutzt eine Client/Server-Architektur. *X* selbst läuft als Server auf dem System, der die Anzeige verwaltet. Client-Programme kommunizieren mit dem *X*-Server und teilen ihm mit, was es zu wo und wann darzustellen gilt.

> „Heute braucht man kaum noch Konfigurationsdateien."



Werkzeuge wie die Anzeigeeinstellungen von KDE helfen beim Einrichten von Multi-Monitor-Systemen. Für ein System mit einem einzigen Bildschirm ist jedoch meist jede Konfiguration unnötig.

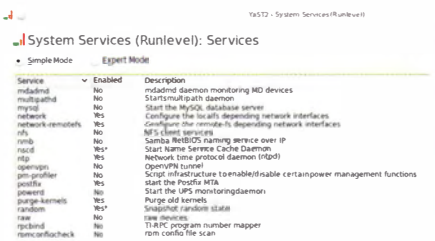
Kompatibilitäts-Features

Das mag nun außerordentlich komplex erscheinen, aber das System nutzt lokale Sockets zur Kommunikation zwischen Client und Server, sodass es sich kaum negativ auf die Geschwindigkeit auswirkt. Ein klarer Vorteil dieser Methode ist aber, dass der Client und der Server nicht auf demselben System laufen müssen. Sie können sich mit einem anderen Computer über SSH verbinden und – das Vorhandensein entsprechender Berechtigungen vorausgesetzt – am entfernten Computer ein Programm starten, dessen GUI auf dem lokalen Computer angezeigt wird. Das ist ein großer Unterschied im Vergleich zu VNC, weil nur ein Anwendungsfenster lokal angezeigt wird und dieses auch nur lokal auf diesem Computer existiert. Eine VNC-Verbindung spiegelt hingegen den gesamten Desktop auf beiden Computern.

Manche schätzen die Client/Server-Architektur als zu kompliziert ein, weshalb es Bestrebungen gibt, ein einfacheres Grafiksystem zu entwickeln. Das am meisten fortgeschrittene Projekt ist *Wayland*. *Wayland* nutzt eine andere Herangehensweise: Nicht nur der Client/Server-Aufbau fehlt, *Wayland* überlässt auch das Rendern der Fenster und anderer Elemente auf dem Bildschirm Programmen wie *OpenGL* oder *Cairo*. Das macht *Wayland* wesentlich einfacher. *X* beinhaltet eine große Menge Code, der der Kompatibilität dient und von der *X*-Spezifikation zwar gefordert, aber nie genutzt wird. Indem die Kontrolle an die Clients übergeben wird, kann *Wayland* viel schlanker, effizienter und zukunftssicherer sein. Das bedeutet auch, dass die grafische Software viel mehr Einfluss auf die Darstellung der GUI hat.

Dämonenprozesse

Wenn Sie den Splash-Screen aktueller Distributionen deaktivieren, werden sie bei den meisten einen Bildschirm voller Informationen darüber sehen, welche Dienste gerade starten. Aber was sind diese Dienste, was machen sie, und sind sie überhaupt alle notwendig? Dienste sind Programme, die im Hintergrund laufen. Einige kümmern sich um das Netzwerk, andere managen die Hardwareerkennung und ihre Konfiguration. Weit mehr Dienste sind aber traditionelle Software, auch Daemons oder Dämonen genannt. Sie stellen anderen Programmen Funktionen zur Verfügung, falls diese gebraucht werden. Um auf die Frage der Notwendigkeit aller Dienste zurückzukommen: Nein, meist werden sie nicht benötigt. Während einige von diesen Diensten auf fast allen Systemen gebraucht werden, so wie der **syslog**-Dämon, der die Log-Informationen des Systems in die entsprechenden Dateien schreibt, werden andere oft nicht gebraucht. So gibt es beispielswiese keinen Grund dafür, CUPS zu starten, wenn sie keinen Drucker verwenden. Ebenso dürfte der MySQL-Datenbankserver meist nicht gebraucht werden, genauso wenig wie der SSH-Dämon, wenn sich Ihr Computer nur alleine in einem Netzwerk befindet. Eine halbe Stunde des Herumexperimentierens könnte Ihnen also eine Sekunde beim Einschalten des PCs ersparen. Sie können sich auch einige Ressourcen sparen, indem Sie unnötige Dienste nicht starten. Sie verbrauchen aber auch kaum Speicher, selbst wenn sie geladen wurden. Und selbst dieser Speicher wird wieder frei, wenn die Dienste aufgrund ihrer Untätigkeit in den Swap-Speicher ausgelagert werden. Deshalb sollten Sie nur jene Dienste deaktivieren, die Sie sicher niemals benötigen werden. Es ist sinnvoll, die nötigen Dienste aktiviert und sie geduldig ihren Netzwerk-Port oder -Socket abhören zu lassen, da das die Arbeit der Client-Software etwas effizienter macht. So müssen Programme beispielsweise keinen eigenen Code besitzen, um Log-Dateien öffnen, beschreiben und wieder schließen zu können: Sie rufen einfach die **syslog()**-Funktion mit dem Log-Eintrag als Parameter auf, und der Dämon kümmert sich um den Rest. **Syslog** ist ein wichtiger Dienst – wenn irgendwas auf dem System schief läuft, ist das oft der erste Ort, um nachzusehen, da die meisten Programme ihre Fehlerinformationen zum System-Log senden (normalerweise unter **/var/log/messages** zu finden). Warum aber werden Hintergrunddienste als Dämonen bezeichnet? Dafür gibt es eine Reihe von Theorien. Wir mögen die Erklärung am liebsten, dass Dämonen in der griechischen Mythologie Geschöpfe waren, die sich um jene Belange kümmerten, für die man keine Gottheit belästigen würde.



Es ist möglich, die Startzeit des Computers durch das Ausschalten von unnötigen Prozessen zu verkürzen.

Netzwerke

Wie Ihr Computer mit anderen spricht.

Vernetzung ist eine der Kernfähigkeiten von Linux. Sogar auf einem Einzel-Computer ohne Verbindung zu einem lokalen Netzwerk werden die Netzwerkfähigkeiten verwendet. Viele Dienste nutzen ein Client/Server-Modell, bei dem der Server im Hintergrund läuft und auf die Anweisungen von anderen Programmen wartet. Sogar so etwas Grundlegendes wie der System-Logger läuft als Netzwerkdienst und ermöglicht es anderen Programmen so, Log-Dateien zu schreiben. Das *X*-Grafiksystem ist ebenfalls so aufgebaut: Der *X*-Server arbeitet im Hintergrund, während ihm die Programme mitteilen, was er anzeigen soll. Das ist der Grund, warum es so einfach ist, *X*-Programme auf einem entfernten PC darzustellen – soweit es das System angeht, gibt es keinen wirklichen Unterschied zwischen diesem Fall und einem Fenster, das nur lokal angezeigt werden soll.

Wenn sie **ifconfig** ausführen, wird zumindest eine Netzwerkschnittstelle mit dem Namen lo angezeigt. Diese besitzt die Adresse **127.0.0.1** – sie wird vom Computer verwendet, um mit sich selbst zu sprechen. Während diese Praxis bei Menschen Bedenken hervorruft, ist sie bei Computern gängige Praxis. Fast alle anderen Vorgänge rund um das Netzwerk basieren auf TCP/IP, entweder über verkabeltes Ethernet oder drahtlos. Aber es gibt noch eine ganze Reihe andere genutzte Netzwerktypen.

Alle Distributionen und Desktops beinhalten gute Werkzeuge, um TCP/IP-basierte Netzwerke zu konfigurieren und zu verwalten, vom allgegenwärtigen *NetworkManager* bis hin zu individuellen Tools wie Gnomes Netzwerk-Konfigurationswerkzeug oder Yast von OpenSUSE.

Aktuelle Dreingaben im Netzwerkbereich kümmern sich um drahtlose 3G-Kommunikation und PAN-Technologien (Personal Area Network) wie etwa Bluetooth. Das Verwenden eines Breitband-Dongles für das 3G-Netz ist recht einfach, meist reicht es dazu aus, *NetworkManager* oder die PPP-Software Ihres Desktops zu verwenden. Ja, 3G-Modems arbeiten wirklich wie Modems, mit Wählskripten und allem Drum und Dran, aber ohne diese quälenden und schmerzhaften Geräusche beim Verbinden (jüngere Leser können diese Aussage ignorieren).

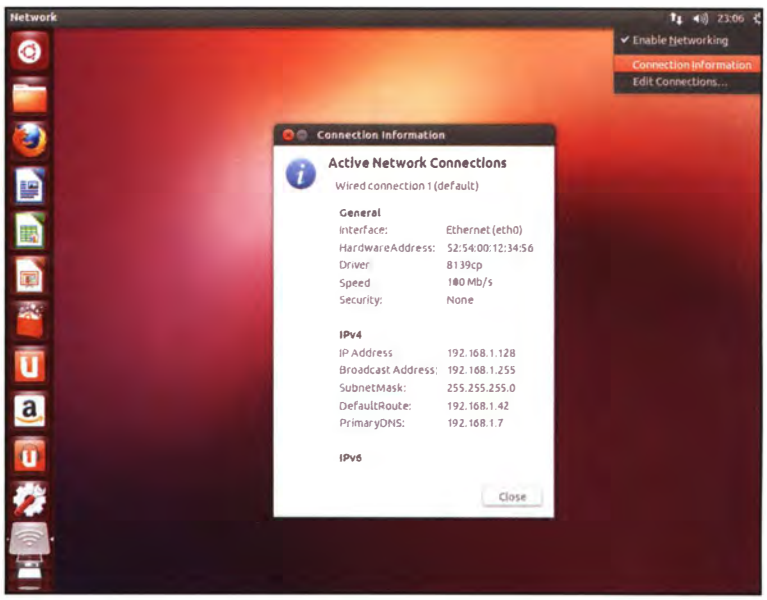
Das Protokoll der Könige

Bluetooth wird mit der steigenden Anzahl von Mobilgeräten immer wichtiger, und auch die Menge der Eingabegeräte, die das Protokoll benutzen, wird immer länger. Es sind nicht nur die Smartphone- und Tablet-Benutzer, die davon profitieren – eine Bluetooth-Maus und Funk-Lautsprecher können auch den Umgang mit einem Notebook verbessern, ohne lange alles zu verkabeln zu müssen, um einsatzbereit zu sein. PulseAudio macht es einfach, da es zwischen den Ausgabegeräten umschalten kann, sobald sie erkannt wurden.

Speicher

Das Speichern von Daten auf einer Festplatte kann mehrere Schichten einbeziehen. Alle physikalischen Speicher (im Gegensatz zu netzwerkbasiertem Speicher) drehen sich bei Linux um blockbasierte Geräte. Ein blockbasiertes Gerät wie **/dev/sda1** verweist wirklich auf Blöcke, bestimmte physisch vorhandene Bereiche auf der Festplatte, die zu einer Partition zusammengefasst werden. Darauf setzt das Dateisystem auf, das bestimmt, wie die gespeicherten Daten in Strukturen bestehend aus Dateien und Verzeichnissen organisiert werden. Diese bestehen wiederum jeweils aus Daten und Metadaten.

Wo liegt der Unterschied zwischen Daten und Metadaten? Angenommen, wir haben eine Datei, die etwas Text enthält. Die Daten in dieser Datei entsprechen dem Text, aber die Datei hat auch Attribute. Dazu gehören der Besitzer, die Zeit der Erstellung, die Zeit, zu der die Datei zuletzt geändert wurde, die Zeit des letzten Zugriffs sowie Informationen darüber, wer die Datei lesen und ändern darf. Das sind die Informationen, die angezeigt werden, wenn Sie den Befehl ls -l auf eine Datei anwenden oder sich ihre Eigenschaften im Datei-Manager ansehen. Das Standard-Dateisystem, das heute verwendet wird, ist ext4, aber es gibt auch alternative Dateisysteme wie ext3, ReiserFS, XFS, JFS und natürlich auch FAT und NTFS aus der Windows-Welt.



› **Das Netzwerk ist unerlässlich für den Betrieb eines Linux-Systems. Das Localhost-Interface wird automatisch eingerichtet, für den Rest haben wir Programme wie den *NetworkManager*.**

Andere Linux-Betriebssysteme

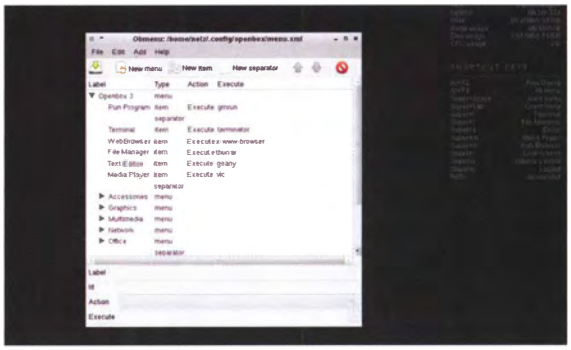
Linux ist nicht nur ein Betriebssystem, das auf „klassischer Hardware" (Desktops, Notebooks, Server) läuft, sondern wird noch in ganz anderen Umgebungen eingesetzt, darunter viele Embedded-Geräte wie Router, Videorecorder oder Set-Top-Boxen. Falls etwa Ihr Router den Zugriff über SSH erlaubt, werden Sie sich womöglich schnell zu Hause fühlen, wenn Sie sich einloggen. Es gibt aber noch eine andere Klasse von Geräten, wo Linux genutzt wird: Smartphones und Tablets, die Android verwenden. Android ist nämlich eigentlich Linux, es nutzt den Linux-Kernel. Aber es ist nicht GNU/Linux. Der Kernel mag zwar auf dem gleichen Source-Code basieren, aber alles, was darüber läuft, ist anders. Die Funktionsprinzipien sind sich in gewisser Weise ähnlich, aber deren Ausführung unterscheidet sich stark – auch wenn Sie ähnliche Kommandozeilenwerkzeuge finden werden, falls Sie ein Terminal-Fenster auf Ihrem Smartphone aufbekommen.

Desktops

Gnome, KDE, Cinnamon, Unity – die Benutzerschnittstelle.

Wenn der Kernel die tiefste Schicht des Systems darstellt, ist das Benutzerinterface die höchste. Alles andere, vom Kernel über die Treiber bis hin zu den Hardwareschnittstellen, ist nutzlos, wenn wir den Computer nicht benutzen können. Allgemein meint man mit dem Benutzerinterface einen grafischen Desktop. Hier begegnen wir dann weiteren Schichten. *X* (oder zukünftig vielleicht *Wayland*) bieten nur eine blanke Fläche. Sie brauchen noch etwas, um Ihnen die Vorzüge eines fensterbasierten Interface zu bieten. Und dieses Etwas ist der Fenstermanager.

In der Vergangenheit waren Fenstermanager Standalone-Systeme. Von diesen gibt es immer noch eine ansehnliche Anzahl, so wie etwa *OpenBox* oder *Enlightment*. Heutzutage sind sie aber oft Teil einer großen Desktop-Umgebung. Um genau zu sein, handelt es sich bei einem Fenstermanager um ein Programm, welches das Öffnen, Schließen, die Positionierung und andere Manipulationen rund um ein Fenster verwaltet. Über die Zeit wuchsen sie, sodass Sie noch weitere Funktionen übernahmen, so wie Taskleisten oder Menüs zum Starten von Programmen. Bis schließlich daraus die heute bekannten Desktop-Umgebungen wurden.



› **Es gibt auch eine große Anzahl von sehr schlanken Fenstermanagern. *OpenBox*, das hier gerade auf CrunchBang läuft, ist einer davon.**

Softwaresammlungen

Eine Desktop-Umgebung ist im Großen und Ganzen eine kombinierte Sammlung von Werkzeugen, die nötig sind, um einen kompletten Desktop zu betreiben. Es laufen Programme darauf, sie manipulieren ihre Fenster, verfolgen, was sich auf dem System abspielt. Im Hintergrund läuft aber immer noch ein Fenstermanager: *KWin* bei KDE und *Metacity* bei Gnome, um nur zwei zu nennen. Es ist die Stufe der Integration, die eine Desktop-Umgebung von einem Fenstermanager unterscheidet. In KDE, wo alles rund um einen gemeinsamen Kern arbeitet und wo Programme nicht nur untereinander kommunizieren, sondern eines auch in das Fenster eines anderen eingebunden werden kann, ist das klar zu erkennen.

Während es nicht viel Sinn ergeben mag, *KWin* mit Gnome zu benutzen, möchten sie vielleicht einen der besseren Fenstermanager ausprobieren, wenn es um das Verwalten von Fenstern geht. Oder vielleicht möchten Sie auch eine andere Art zum Anzeigen von Fenstern verwenden. Beispielsweise gibt es Fenstermanager, die auf das sogenannte Tiling setzen, wie etwa *awesome* und *xmonad*. Diese skalieren die Fenster automatisch so, dass sie alle auf den Desktop passen. KDE hat eine eigene Option, um dieses Verhalten zu erreichen. Es gibt Fenstermanager, die darauf ausgelegt sind, mit der Tastatur steuerbar zu sein, genauso wie solche Exemplare, die auf sehr spezialisierten Systemen von Nutzen sind. Diese führen Programme im Vollbildmodus aus und wollen nicht, dass irgendein Widget Platz verschwendet.

› **Gnome, KDE, Unity, Cinnamon, Mate – die Auswahl ist nicht gerade klein, wenn es um die passende Desktop-Umgebung geht. Aber hat von Ihnen jemand mehr als eine Handvoll ausprobiert?**

Sound

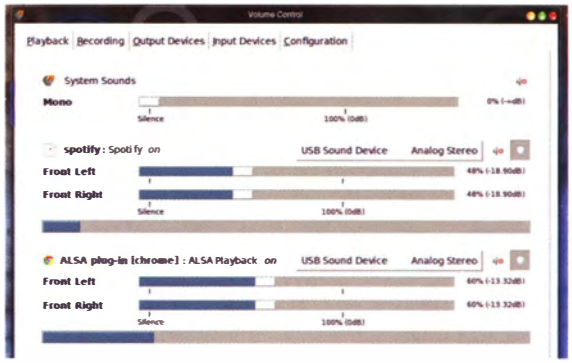
Einst ein heikles Thema.

Das Soundausgabesystem unter Linux hat schon einige Anpassungen hinter sich. Über Jahre war OSS (Open Sound System) Standard, bevor es durch ALSA (Advanced Linux Sound Architecture) abgelöst wurde. ALSA gibt es immer noch, inzwischen gesellten sich aber auch PulseAudio und Jack dazu.

ALSA unterstützt mehrere Soundkarten, wovon jede mehrere Ein- und Ausgänge haben kann. Sofern es die Hardware unterstützt, beherrscht es auch Hardware-Mixing. Andernfalls nutzt es stattdessen Software-Mixing. Damit entfällt die Notwendigkeit eines Soundmanagers, wie er in der Vergangenheit von KDE und Gnome zur Verfügung gestellt wurde. Ein Soundmanager wurde gebraucht, damit mehr als ein Programm gleichzeitig Sound abspielen kann. ALSA arbeitet sehr hardwarenah, um die Latenz möglichst gering zu halten. Inzwischen wird fast jede Hardware direkt unterstützt. Die Musikwiedergabe sollte also sofort nach der Installation der Distribution funktionieren. ALSA ist eine Kombination aus Kernel-Code und User-Space-Anwendungen. Es stellt auch eine API zur Verfügung, sodass es von anderen Anwendungen direkt kontrolliert werden kann. Dazu gehören etwa die Lautstärkeregler, die in den meisten Desktop-Umgebungen integriert sind.

PulseAudio-Performanz

PulseAudio ist ein neueres Audio-Framework, aber kein Ersatz für ALSA. Stattdessen setzt es auf dem Kernel-Audio-System auf und erlaubt so eine erweiterte Kontrolle darüber. PulseAudio arbeitet als Server, der Eingaben von den Quellen annimmt und sie an die Ausgabe (Hardware oder eine Aufzeichnungssoftware) weiterleitet. In vielen Fällen befindet sich ALSA auf der Ausgangsseite, aber auch am Eingang kann ein ALSA-Treiber aktiv sein. Das kommt beispielsweise dann vor, wenn eine Anwendung PulseAudio nicht direkt unterstützt. Das Funktionsprinzip



› **Falls Ihnen jemand klarmachen will, wie kompliziert PulseAudio ist, ist es am besten, nicht mit ihm zu diskutieren. Als ob die Gestaltung dieses Fensters jemanden groß interessieren würde, solange es funktioniert.**

lässt sich so weit treiben, dass ein Ausgangsignal, das von einer Anwendung an ALSA gesendet wurde, von eben jenem abgefangen und an PulseAudio geleitet werden kann, welches es wieder zurück zu ALSA sendet. Deshalb ist es keine Überraschung, dass viele Anwender PulseAudio für übertrieben kompliziert halten. Ein gutes Interface sollte dem Benutzer eine solche Verkettung aber transparent machen. Die meisten Distributionen erfüllen diese Anforderung auch und bieten so zusammen mit ALSA eine funktionierende Soundlösung, die auch mit mehreren Soundausgabegeräten zusammenarbeitet. ALSA unterstützt zwar mehrere Ausgabegeräte, das Standardgerät ist aber systemweit als solches festgelegt. PulseAudio unterstützt es dagegen, die Musik direkt zu den Lautsprechern zu leiten, während sie das ebenfalls mit dem PC verbundene Bluetooth-Headset daran vorbei gerade für einen VOIP-Anruf verwenden. Des Weiteren ist es möglich, die Lautstärke einzelner Programme separat zu regeln. PulseAudio ist außerdem netzwerkfähig. Es kann dazu verwendet werden, andere PulseAudio-Server zu finden und über deren Lautsprecher Sound abzuspielen.

JACK (Jack Audio Connection Kit) ist ein Sound-Server, der für professionelle Audio-Anwendungen entworfen wurde. Seine Stärke ist eine latenzarme Echtzeitverbindung zwischen Anwendungen speziell für Sound- und MIDI-Daten. Für den täglichen Computergebrauch ist die Software unnötig, aber sehr nützlich für angehende Musiker.

Drucken

CUPS und Treiber.

Während Open Source Wahlfreiheit fördert und deshalb oft auch mehrere Programme die gleiche Aufgabe auf leicht unterschiedliche Art und Weise erledigen, gibt es einige Bereiche, wo bestimmte Programme im Prinzip konkurrenzlos dastehen. Was *X.org* für die universelle Grafikdarstellung darstellt, ist CUPS für den Bereich des Druckens.

Wenn Sie einen Drucker an Ihrem Linux-Rechner angeschlossen haben, brauchen Sie zwei Dinge: CUPS und einen Treiber für Ihren Drucker. In vielen Fällen erhalten Sie beides gemeinsam. CUPS ist ein Server, der im Hintergrund auf Druckanfragen wartet. So gut wie jede Anwendung, die das Drucken beherrscht, kennt auch das Internet Printing Protocol (IPP), dessen Sprache auch die von CUPS ist. Wenn Sie in Ihrer Textverarbeitung oder im Browser den Drucken-Button betätigen, erscheint ein Fenster, das Ihren Drucker zeigt, zusammen mit einer Auswahlmöglichkeit, falls Sie mehrere davon besitzen. Die Anwendung muss nur die zu druckenden Daten senden, was normalerweise in PostScript geschieht, CUPS kümmert sich um alles Nachfolgende. Auch um die an Sie gerichtete Hinweismeldung, dass Sie zum Drucken doch bitte den Drucker einschalten möchten. CUPS muss dabei nicht einmal auf dem lokalen Computer laufen. Es ist ein netzwerkbasierter Dienst und macht so den Drucker am lokalen PC auch für jeden anderen Rechner im Netzwerk verfügbar. Ohne Treiber, der CUPS sagt, wie mit dem Drucker gesprochen wird, ist es aber nutzlos. Deshalb bringt es eine große Anzahl von Treibern mit. Viele weitere gibt es, wenn Sie das **gutenprint**-Treiberpaket installieren (und zwar so viele, dass Ihre Distribution dieses Paket bereits standardmäßig installiert haben dürfte). Von HP gibt es ein Treiberpaket namens **hplib**, das sowohl die Drucker wie auch die Scanner des Herstellers abdeckt.

Manche Hersteller bestehen aber darauf, ihre eigenen Treiber anzubieten, anstatt sie mit CUPS zu bündeln. Meist geht es dabei um Lizenzierungsfragen. In diesem Fall haben Sie die Alternative, die Internetseite des Druckerherstellers auf einen passenden Treiber hin zu durchsuchen und diesen separat zu installieren. Der Drucker sollte anschließend unter CUPS und im Konfigurationsfenster ihrer Distribution auftauchen. Sie können auch vor dem Kauf eines Druckers auf **linuxprinting.org** vorbeischauen, damit Sie zu einem Modell eines kooperativeren Druckerherstellers greifen können. ■

Lernen Sie Linux

MIT MIKE SAUNDERS

Lektion 1: Sie wollen einen Job, bei dem Sie mit Linux arbeiten? Dann nehmen Sie gleich in der ersten Reihe Platz und besorgen Sie sich die Informationen, die Sie für die LPI-Zertifizierung brauchen. Los geht's mit: Hardware.

Mike

Linux Professional Institute

Unser Experte

Seit über zehn Jahren veröffentlicht Mike Saunders Artikel über Linux. Er hat in seinem Leben vermutlich mehr Distributionen installiert als warme Abendessen zu sich genommen.

So, Sie sind also Linux-Experte? Freunde und Verwandte glauben das sicher gern, sobald Sie ihnen ein paar Kunststückchen an der Kommandozeile gezeigt haben. Aber Ihr Arbeitgeber in spe ist womöglich nicht so leicht zu beeindrucken. Hier ist es eine gute Idee, sich das eigene Fachwissen von einer kompetenten Stelle zertifizieren zu lassen.

Zertifizierung

In der Linux-Welt haben wir glücklicherweise eine ausgezeichnete Einrichtung, die sich um genau dieses Problem kümmert: die LPI-Zertifizierung. LPI ist die Abkürzung für das Linux Professional Institute, eine Non-Profit-Organisation, bei der man Qualifizierungskurse absolvieren und Prüfungen über Linux-Systeme ablegen kann. Es gibt drei Ebenen der Zertifizierung. In der ersten geht es um allgemeine Systemadministration einschließlich Hardware-Konfiguration, Befehlszeilen-Operationen, Paketmanagement und Prozesse.

In diesem Teil erklären wir Ihnen alles, was Sie für die Prüfung LPI 101 wissen müssen. Bei dieser Prüfung wird getestet, ob Sie sich in einem geschäftlichen Umfeld professionell um die Linux-Rechner kümmern können. Wenn Sie sich schon eine Weile mit Linux beschäftigen, dann wird Ihnen dieser Teil vermutlich bekannt vorkommen. Lesen Sie ihn sich trotzdem durch. Vielleicht findet sich doch etwas, das Sie noch nicht wussten, und dann hat es sich gelohnt, die Seiten durchzuackern.

Empfohlene Linux-Distribution

An dieser Stelle sei angemerkt, dass die LPI-Studientexte sich größtenteils auf langlebige Linux-Distributionen beziehen, bei denen nicht alle sechs Monate eine komplett neue Version erscheint. Red Hat Enterprise Linux (RHEL) ist ein gutes Beispiel, aber RHEL ist nicht gerade billig. CentOS – ein Nachbau von RHEL – kann man hingegen gratis bei **www.centos.org** herunterladen. Es ist eine ausgezeichnete Basis für die Schulung. Wir verwenden hier die Version CentOS 5.5. Eine andere, ebenfalls gute Alternative ist Debian, das sich an die Standards hält und über Jahre hinweg stabil bleibt.

Aber genug der Vorbemerkungen, fangen wir an! In dieser Lerneinheit steht die Hardware im Zentrum. Schritt für Schritt werden Sie mit unserer Unterstützung herausfinden, welche Geräte in Ihrem System sind, wie Sie Treiber laden beziehungsweise deaktivieren und vieles mehr.

Teil 1: Auflisten der Hardware

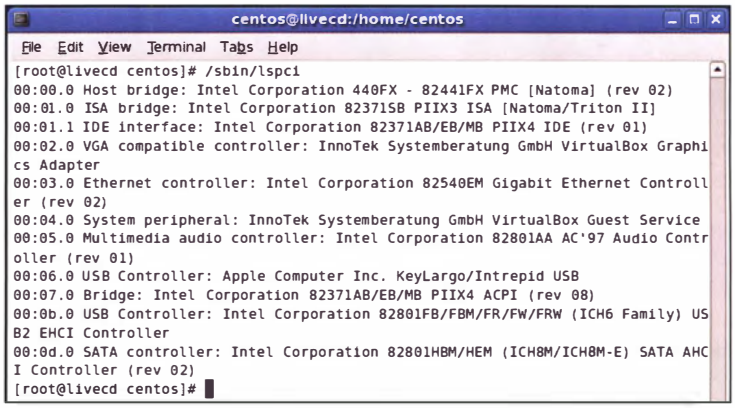
Selbst im günstigsten Fall sind PCs immer noch ziemlich komplizierte Maschinen. Sie geben sich zwar alle Mühe, cool und modern auszusehen, doch unter der Oberfläche finden sich noch immer etliche Überbleibsel aus den 1980er Jahren. Zum Glück gibt es offene Systeme wie Linux, die uns Möglichkeiten bieten, wie wir an viele Informationen über Laufwerke und Peripheriegeräte herankommen können. Die Startpunkte dafür sind die Verzeichnisse **/proc** und **/sys**. Dies sind keine „echten" Ordner wie beispielsweise **home**, sondern virtuelle Verzeichnisse, die das System automatisch anlegt und die Informationen über die Betriebsabläufe und die Hardware enthalten. Wozu braucht man diese Verzeichnisse? Bei **/proc** geht es vor allem um Informationen über die Betriebsabläufe (also die Software-Programme, die gerade auf dem System laufen), und **/sys** deckt vorrangig die Hardware ab. Allerdings gibt es ein paar Überschneidungen.

Der Befehl lscpi

Die meisten internen Komponenten Ihres PCs sitzen auf einem Datenübertragungssystem, dem sogenannten PCI-Bus.

Bei älteren Distributionen erhalten Sie Informationen über die Laufwerke über den Befehl **cat /proc/pci**, doch diese Datei existiert in neueren Distributionen nicht mehr. Stattdessen können Sie Ihr Glück mit **/sys/bus/pci/devices** versuchen, allerdings sind die entsprechenden Informationen eigentlich nicht für Normalsterbliche gedacht. Wir benutzen besser die Befehlszeile. Öffnen Sie dazu zunächst ein Terminal über Anwendungen > Zubehör > Terminal, und wechseln Sie zum Root-User (Admin), indem Sie **su** eingeben. Geben Sie dann folgenden Befehl ein:

```
/sbin/lspci
```



› Die Ausgabe des lspci-Befehls zeigt die Hardware-Komponenten im Innern des Rechners an.

Sie erhalten daraufhin eine Liste aller Hardware-Komponenten auf dem PCI-Bus Ihres Rechners. Wie das aussieht, sehen Sie in der Grafik rechts. In der Liste sollten Ihre Videokarte, der Ethernet-Adapter und andere Geräte auftauchen. Genauere Einzelinfos erhalten Sie, wenn Sie an den Befehl noch ein **-vv** hängen. Die Ausgabe zeigt Ihnen Informationen über Interrupt-Verbindungen und I/O (Input/Output)-Anschlüsse. Mithilfe eines Interrupts kann ein Gerät der CPU mitteilen, dass Handlungsbedarf besteht – wenn zum Beispiel eine Soundkarte dem Betriebssystem mitteilt, dass sie eine Operation abgeschlossen hat –, und I/O-Anschlüsse dienen dazu, Daten von und zu einer Komponente zu senden.

Ein PC verfügt nur über eine begrenzte Anzahl von Interrupt-Verbindungen (abgekürzt: IRQs). Früher, als die meisten Systeme gerade mal einen Monitor und eine Tastatur ihr Eigen nannten, war das vollkommen ausreichend, aber heutzutage natürlich nicht mehr. Um das Problem zu umgehen, können moderne Geräte sich die IRQs teilen. Man muss mittlerweile also nicht mehr an den IRQs und I/O-Anschlüssen herumfrickeln, doch mit dem vorher erwähnten Befehl **lspci -vv** können Sie jederzeit eine genaue Liste der Hardware-Komponenten aufrufen, die gerade in Benutzung sind.

Teil 2: Treibermodule

Wie gehen Sie vor, wenn Sie ein Hardware-Element deaktivieren wollen? Nun, zuerst müssen wir herausfinden, wodurch das Element überhaupt aktiviert wird: den Hardware-Treiber nämlich. Bei Linux gibt es zwei Möglichkeiten, wie Treiber bei der Kompilation des Betriebssystemkerns oder Kernels aktiviert werden können. Wer die Distribution zusammenstellt, kann die Treiber entweder direkt in den Kernel kompilieren oder als ein alleinstehenden Modul, das der Kernel bei Bedarf lädt. Die zweite Variante ist bei Weitem die üblichere, weil dadurch der Kernel kleiner bleibt, was den Systemstart verkürzt und das Betriebssystem deutlich flexibler macht.

Sie finden Ihre Module im Verzeichnis **/lib/modules/<kernelversion>/kernel**. Dies sind die **.ko** (kernel object)-Dateien, und Sie können sehen, dass sie in einzelnen Verzeichnissen für Sound, Dateisysteme und anderes sortiert sind. Wenn Sie nun in das Treiber-Unterverzeichnis **drivers** gehen, finden Sie dort noch mehr Modulkategorien. Wichtig ist hier die Unterscheidung zwischen block- und zeichenorientierten Geräten – im Englischen als *block devices* und *char devices* bezeichnet. Blockorientierte Geräte sind Hardware, bei der Daten in großen Datenblöcken übertragen werden, wie zum Beispiel Festplatten. Zeichenorientierte Geräte wie Mäuse und serielle Anschlüsse hingegen übertragen die Daten jeweils nur byteweise nacheinander.

Der Linux-Kernel ist intelligent und kann Module laden, sobald er bestimmte Hardware-Komponenten entdeckt. Er ist sogar so clever, dass er bei Bedarf Module lädt, wenn ein USB-Gerät angeschlossen wird – aber zu USB kommen wir später. Vorher wollen wir uns noch anschauen, wie man Module organisiert. Um eine Liste von allen Modulen zu erhalten, die der Kernel im Moment geladen hat, geben Sie diesen Befehl als Root-Befehl ein:

```
lsmod
```

Bei manchen Distributionen, so auch bei CentOS, müssen Sie noch **/sbin/ - ie /sbin/lsmod** voranstellen. Es erscheint eine Liste, deren genauer Inhalt bei verschiedenen Systemen unterschiedlich ausfällt, je nachdem, welche Hardware Sie installiert haben.

Modulnamen

Bei einigen der Module wird Ihnen vom Namen her sofort klar sein, worum es sich handelt, etwa bei **cdrom** und **battery**. Bei bestimmten Modulen sehen Sie in der rechten Spalte eine Liste von „Used By"-Modulen. Diesen entsprechen in der Welt des Paketmanagements die Abhängigkeiten. Die „Used By"-Spalte zeigt, welche Module darauf angewiesen sind, dass andere vor ihnen geladen werden.

Aber was ist mit Modulen, die kryptische Namen haben? »

Quick-Tipp

Bestimmte Geräte wie etwa PS/2-Mäuse und -Tastaturen dürfen nur an den Computer angeschlossen werden, wenn dieser ausgeschaltet ist. Werden sie bei laufendem Betrieb ein- oder ausgesteckt, kann dies die Chips auf dem Motherboard beschädigen.

Was ist /dev?

Alles ist eine Datei – so lautet jedenfalls einer der Leitsätze von Unix und somit auch von Linux. Diese Aussage bezieht sich nicht nur auf Ihre Dokumente und Bilder, sondern auch auf die Hardware. Zuerst mag das seltsam klingen – wie kann eine Hardware-Komponente als Datei dargestellt werden? In einem weiteren Sinne ist eine Datei etwas, aus dem man Informationen lesen und in das man Informationen schreiben kann, und das trifft durchaus auch auf eine Festplatte zu. Allerdings gibt es auch Geräte wie etwa einen Zufallsgenerator, die nur in eine Richtung funktionieren: Man kann sie lesen, aber nichts zurückschicken. Das **/dev**-Verzeichnis enthält Hardware-Geräteknoten – das sind Dateien, die die Geräte repräsentieren. Zum Beispiel ist **/dev/dvd** Ihr DVD-ROM-Laufwerk. Wenn eine DVD im Laufwerk liegt, können Sie **cat /dev/dvd** eingeben, dann schickt das Laufwerk seine Binärdaten zu Ihrem Terminal. Geräteknoten werden automatisch vom System angelegt, manche werden in Unterverzeichnisse wie **snd** (Soundkarten) oder **input** (Mäuse) verschoben. Das virtuelle Gerät **/dev/null** frisst einfach nur Daten und zerstört sie – dorthin kann etwa Output umgelenkt werden, der nicht auf dem Monitor erscheinen soll.

» Was genau machen die? Hier hilft Ihnen der **modinfo**-Befehl. Mit dem Kommando

```
/sbin/modinfo dm_mod
```

beispielsweise rufen Sie Informationen zum Modul **dm_mod** ab, darunter eine Menge technische Daten, aber auch eine kurze Angabe der Funktion des Moduls. Diese Kurzbeschreibung ist für die meisten – wenn auch leider nicht alle – Module vorhanden.

Wie erwähnt, viele Module werden automatisch vom Kernel geladen. Sie können das Laden aber auch durch den **modprobe**-Befehl erzwingen. Dieses kleine Dienstprogramm ist sowohl für das Laden als auch das Entladen von Modulen aus dem Kernel verantwortlich und leistet gute Dienste, wenn man bestimmte Kernelfunktionen auf die Schnelle aktivieren oder deaktivieren möchte. Zum Beispiel sehen Sie auf unserer Liste die Module **lp, parport** und **parport_pc**. Diese betreffen Drucker, die an der Parallelschnittstelle angeschlossen werden. Solche Drucker werden heute kaum noch eingesetzt, daher können wir diese Module deaktivieren und dadurch ein bisschen RAM freimachen. Doch woher wissen Sie, in welcher Reihenfolge die Module eingegeben werden müssen? Sie können es erschließen, wenn Sie sich die vorhin erwähnte „Used By"-Spalte anschauen. Stellen Sie das Modul, von dem die beiden anderen abhängen, an das Ende der Befehlszeile. Zuerst wird dann das **lp**-Drucker-Modul entfernt, dann das Modul für PC-spezifische Parallelport-Drucker **parport_pc** und schließlich der generisch Parallelport-Treiber **parport**. Der korrekte Befehl lautet demnach:

```
/sbin/modprobe -r lp parport_pc parport
```

Quick-Tipp

Informieren Sie sich über die LPI-Zertifizierung auf der Website des Instituts: **www.lpi.org**. Dort finden Sie eine ausführliche Liste der Lernziele, um die es in den Qualifizierungskursen geht, sowie Beispiele von Prüfungsfragen.

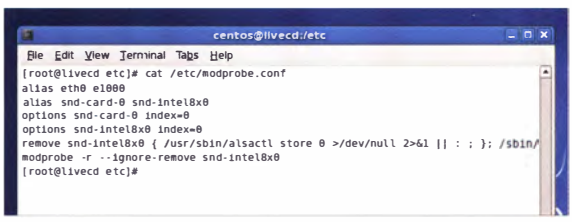
Vereinfachen und vereinheitlichen

Auf ähnliche Weise können wir diese Module durch einen einfachen **modprobe**-Befehl wieder laden (ohne das Entfernungszeichen **-r**). Weil alles in einem System von Abhängigkeiten organisiert ist, brauchen Sie nur das erste Modul aus der Liste zu nennen, und **modprobe** sucht sich dann selbständig, was es sonst noch braucht:

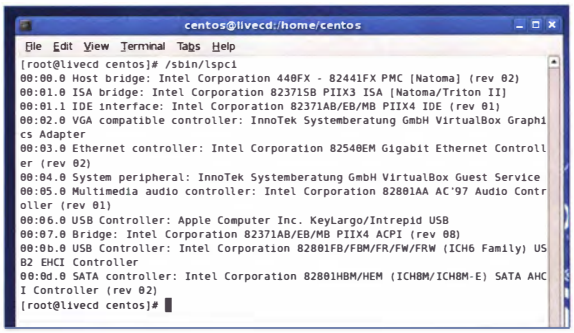
```
/sbin/modprobe lp
```

Dieser Befehl lädt automatisch auch **parport_pc** und **parport**, was wir mit einem schnellen **lsmod**-Befehl überprüfen können.

Auch wenn Linux automatisch und sehr souverän mit Modulen umgeht, kann es manchmal sinnvoll sein, wenn man ein wenig manuellen Input in den Prozess einbringt. Dazu gehen wir in die **/etc/modprobe.conf**-Datei. Als Erstes nehmen wir uns die Aliase vor, mit denen man einen Kurzbegriff für eine ganze Liste von Modulen eingeben kann. Gesetzt den Fall, Sie



› Beispiel einer /etc/modprobe.conf-Datei in CentOS 5.5. Wie man hier in der vorletzten Zeile sieht, können remove-Befehle erweitert werden.



› Eine Liste der aktuell geladenen Module, die mit dem lsmod-Befehl angezeigt wird.

möchten Ihre Soundkarte manuell deaktivieren und aktivieren, aber Sie erinnern sich nicht an alle Module, auf die die Karte zugreift, dann können Sie eine solche **alias**-Zeile einfügen:

```
alias sound snd-ens1371
```

Jetzt brauchen Sie nur **modprobe sound** einzugeben, und Ihre Karte ist aktiviert, auch wenn Sie sich nicht an den spezifischen Treiber erinnern können. Auf diese Weise können Sie die von Ihnen verwendeten Befehle auf verschiedenen Rechnern vereinheitlichen.

Die Funktion **options**, also Optionen, gibt Ihnen die Möglichkeit, Einstellungen an ein Modul zu schicken, um seine Funktionsweise zu konfigurieren. Wenn Sie wissen möchten, welchen Optionen bei einem spezifischen Modul möglich sind, geben Sie wie oben beschrieben den **modinfo**-Befehl ein und durchsuchen Sie die Ausgabe nach **parm** gewidmeten Abschnitten.

Wenn man zum Beispiel den Befehl **modinfo snd-intel8x0** eingibt, erscheint eine Liste von **parm**-Abschnitten, die anzeigen, welche Optionen für dieses Soundchip-Modul zur Verfügung stehen. Eine Option ist index benannt. Unser CentOS auf VirtualBox zeigt diese Option in **/etc/modprobe.conf** an als:

```
options snd-intel8x0 index=0
```

Benutzerdefinierte Befehle

Zu guter Letzt wollen wir noch sogenannte Facilities installieren und entfernen. Diese Zusatzbefehle sind sehr wirkungsvoll: Mit Ihnen können Sie Befehle durch andere ersetzen. So sehen wir zum Beispiel in CentOS auf *VirtualBox* folgenden Befehl:

```
remove snd-intel8x0 { /usr/bin/alsactl store 0...
```

Die vollständige Zeile ist noch viel länger, doch prinzipiell bedeutet das: „Wenn der Benutzer oder das System **modprobe -r snd-intel8x0** eingibt, führe stattdessen den Befehl aus, der mit **alsactl** beginnt – ein Dienstprogramm zur Lautstärkenregelung." Auf diese Weise können Sie auch Lösch- und Protokollierungs-Operationen durchführen, bevor das Modul entfernt wird. Um zu verhindern, dass ein Modul überhaupt geladen wird, belegen Sie es einfach mit dem Alias **off** in **/etc/modprobe.conf**, also:

```
alias parport off
```

Dadurch wird das Modul gar nicht erst geladen, und die entsprechende Hardware wird normalerweise überhaupt nie aktiviert.

Was sind HAL, udev und D-Bus?

Desktop-Umgebungen wie *Gnome* und KDE greifen nicht direkt auf die Hardware zu. Schließlich möchten *Gnome*-Hacker, die eine neue App für die Fotoverwaltung entwickeln, ja keine Programme schreiben, mit denen Daten über ein USB-Kabel in eine Digitalkamera gestopft werden – das soll stattdessen eine tieferliegende Softwareebene übernehmen. Früher war der HAL-Daemon (HAL ist die Abkürzung für Hardware Abstraction Layer) für diese Abstraktion zuständig, doch das Programm wurde durch **udev** ersetzt, einen im Hintergrund laufenden Prozess, der Geräteknoten in **/dev** anlegt und mit der Hardware kommuniziert. Wie kommunizieren Programme mit **udev**? Sie tun das vor allem über D-Bus, ein Softwaresystem für Interprozesskommunikation (IPC), über das Programme Nachrichten austauschen können. Zum Beispiel kann eine Desktop-Umgebung D-Bus auffordern, dass sie immer benachrichtigt wird, wenn ein neues Gerät an den Rechner angeschlossen wird. D-Bus erhält diese Information von **udev**, sobald der Nutzer neue Hardware anschließt, und informiert dann den Desktop, damit dieser eine Dialogfenster öffnen oder ein App starten kann.

Teil 3: USB-Peripherie

PS/2, AUX, seriell, parallel – in den 1990er Jahren benötigte fast jedes Gerät seinen eigenen Stecker, und es herrschte ein riesiges Durcheinander. Zum Glück ist das heute viel einfacher – dank USB! Praktisch jeder normale Computer, der in den letzten zehn Jahren hergestellt wurde, besitzt mindestens einen USB-Anschluss. Und die Entwicklung der Technik ist dort nicht stehen geblieben: USB 2.0 und 3.0 erhöhten die Übertragungsgeschwindigkeit und können inzwischen mit anderen Systemen wie FireWire konkurrieren.

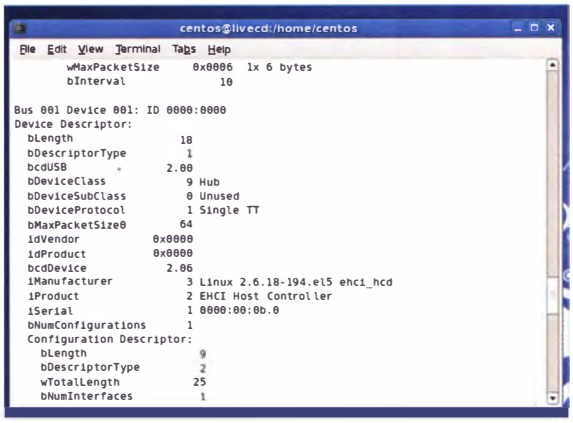
Wenn Sie ein USB-Gerät anschließen, überprüft der Kernel zuerst, zu welcher Klasse es gehört. USB-Geräte werden in Klassen eingeteilt, um die Entwicklung von Treibern zu erleichtern. Es gibt Klassen für Audio-Geräte, Drucker, Webcams oder Benutzerschnittstellen-Geräte (Mäuse, Tastaturen, Joysticks), für bestimmte Geräte sogar händlerspezifische Klassen.

ls + usb = lsusb

Der USB-Support ist bei Linux ausgezeichnet – es gibt viele Hilfsmittel, mit denen ein Administrator herausfinden kann, was hinter den Kulissen vor sich geht. So sitzt der USB-Controller typischerweise auf dem USB-Bus, und wir können unseren verlässlichen **lspci**-Befehl eingeben, wenn wir wissen wollen, welche Art von USB-Controller wir vor uns haben:

```
/sbin/lspci | grep -i usb
```

Hier nehmen wir die Ausgabe von **lspci** und leiten Sie per Pipe-Befehl (mit dem Symbol „|“ gekennzeichnet) an das **grep**-Dienstprogramm weiter, um nach dem Wort USB zu suchen, egal, ob es in Groß- oder Kleinbuchstaben auftaucht. Zum Pipe-Befehl und **grep** kommen wir später, vernachlässigen Sie diese Elemente bitte für den Augenblick. In der Quintessenz filtert diese Befehlszeile die **lspci**-Ausgabe, sodass nur diejenigen Zeilen erscheinen, die Informationen über USB enthalten.



Screenshot der Informationen durch den Befehl lsusb -v.

Wenn Sie den Befehl eingegeben haben, erhalten Sie einige Zeilen mit Informationen zum Händler und zur Art Ihres USB-Controllers. Es ist etwas verwirrend, denn es gibt zwei Typen von Standard-Controllern für USB 1: UHCI und OHCI. Für USB 2.0 wurde wiederum EHCI entwickelt. Die Unterschiede zwischen diesen Controller-Typen können Sie einstweilen vernachlässigen, da der Kernel sich darum kümmert, doch merken Sie sich schon einmal, dass Fragmentierung in der USB-Welt kein Fremdwort ist.

Ähnlich wie **lspci** funktioniert ein Befehl, der Ihnen eine Liste aller Geräte ausspuckt, die an unseren USB-Controller angeschlossen sind. Er lautet:

```
/sbin/lsusb
```

Wenn man den Befehl allein für sich verwendet, bekommt man ziemlich wenig Information – nur eine Liste von Gerätenummern und ihre Position auf dem USB-Bus. Mehr erreichen Sie, wenn Sie **-t** hinzufügen: Die Geräte werden nun in einer Baumstruktur dargestellt, sodass Sie sehen können, welches Gerät mit welchem verbunden ist. Wenn Sie **-v** anhängen, erhalten Sie viel ausführlichere Informationen, wie man auf dem Screenshot in der linken Spalte sehen kann.

Gehen Sie die Ausgabe einmal durch, und Sie werden Information darüber finden, welchen Typ USB-Controller Sie benutzen, und darüber, welche Geräte zurzeit an Ihren Rechner angeschlossen sind. Wenn Sie jetzt noch Lust auf ein kleines Abenteuer haben, dann rufen Sie **/sys/bus/usb/devices** auf, und Sie erhalten Verzeichnisse für jedes Gerät – mit Dateien, die den Herstellernamen, die Geschwindigkeit, den maximalen Stromverbrauch und vieles mehr enthalten.

Wie oben schon besprochen, werden Hardware-Komponenten in Linux normalerweise durch Kernelmodule unterstützt. Das gilt auch für USB. Geben Sie zum Beispiel diesen Befehl ein:

```
/sbin/lsmod | grep hci
```

Auf unserem Rechner wird nun angezeigt, dass der Kernel ein Modul für den OHCI-Controller geladen hat, zusammen mit einem EHCI-Modul zur Unterstützung von USB 2.0.

Nachrichten vom Kernel

Wenn Sie herausfinden wollen, wie der Kernel ein USB-Gerät erkennt, verwenden Sie am besten den **dmesg**-Befehl. Der Befehl liefert Ihnen eine Liste der Nachrichten, die der Kernel seit dem Hochfahren generiert hat. Geben Sie **dmesg** ein und schließen Sie dann ein USB-Gerät an. Warten Sie ein paar Sekunden, bis der Kernel das Gerät erkannt hat, und geben Sie dann noch einmal **dmesg** ein. Sie sehen den Unterschied sofort: Ein paar neue Zeilen erscheinen unten auf der **dmesg**-Ausgabe, die anzeigen, dass der Kernel das Gerät (hoffentlich) erkannt und aktiviert hat. ■

Booten ohne Hardware

Linux wird normalerweise auf Rechnern installiert, die mit der Standardperipherie (Tastatur, Maus, Monitor) ausgestattet sind. Wenn Sie die richtigen Tab-Space-Enter-Kombinationen für den Installer Ihrer Distribution kennen, dann können Sie wahrscheinlich auf die Maus verzichten. Doch ohne die anderen Geräte werden Sie nicht auskommen.

Oder etwa doch? In einer Server-Umgebung, in der Ihr Rechner in einem Gestell einmontiert und schwierig zu erreichen ist, müssen Sie die Installation vielleicht ohne diese Peripheriegeräte vornehmen – und dann wird Booten über das Rechnernetz für Sie interessant. Das Zauberwort bei dieser Methode heißt PXE – Preboot Execution Environment. Dabei handelt es sich um eine vorinstallierte Firmware, die das Netzwerk nach einem NBP (Network Bootstrap Program) durchsucht und es dann lädt und ausführt. Damit diese Methode klappt, benötigen Sie funktionierende DHCP- und TFTP-Server auf Ihrem Netzwerk, wobei der TFTP-Server die passenden Boot-Dateien für die Distribution bereitstellt.

Falls das BIOS Ihres Rechners PXE nicht unterstützt, bleibt Ihnen noch eine weitere Option: USB-Booting. Sie können ein rudimentäres Linux-System von einem USB-Stick aus booten, das dann selbstständig ein umfangreicheres Setup aus dem Netzwerk lädt.

Lernen Sie Linux

MIT MIKE SAUNDERS

Lektion 2: Nach unserem Ausflug in die Hardwarewelt polieren wir in dieser Sitzung Ihre Linux-Kenntnisse mit einem genauen Blick auf den Bootprozess auf.

Mike

Linux Professional Institute

Sie drücken den Einschaltknopf an Ihrem PC. Ein ganze Latte von Nachrichten scrollt über den Bildschirm oder vielleicht auch eine schicke Animation, wenn Sie eine desktoporientierte Distribution verwenden. Schließlich erscheint die Aufforderung zum Login. Doch was passiert eigentlich, bis wir zu diesem Punkt gelangt sind? Darum geht es in der zweiten Lektion der Linux-Schule.

Wie schon in der ersten Lektion, die sich mit der Identifikation und Verwaltung der Hardware auf Ihrem Linux-System beschäftigte, soll Ihnen diese Lerneinheit dabei helfen, sich auf die Zertifizierung durch das Linux Professional Institute (LPI) vorzubereiten. Ob Sie in der Linux-Welt einen Job haben möchten oder ob Sie nur etwas mehr über Ihr Betriebssystem erfahren wollen – diese Lektion lohnt sich auf jeden Fall für Sie.

In der vorigen Lektion haben wir mit CentOS gearbeitet – jetzt ist Debian (Version 5) an der Reihe. Verschiedene Distributionen setzen bestimmte Anwendungen unterschiedlich ein, doch das hier Vermittelte ist zum größten Teil allgemein anwendbar.

Teil 1: Vom Einschalten bis zum Desktop

Der Bootprozess von Linux setzt sich aus einer komplizierten Ansammlung von Betriebsabläufen und Skripten zusammen. Alle gemeinsam machen aus Ihrem PC, der erst einmal nur ein Haufen Metall ist, einen leistungsstarken Rechner oder Server.

BIOS und Bootloader

Das BIOS ist ein kleines Programm, das in einem Chip auf dem Motherboard sitzt. Wenn Sie Ihren PC einschalten, fängt die CPU an, es durchzugehen. Sicher sind Ihnen die „Drücken Sie F2 für Setup"-Nachrichten aufgefallen, die beim Booten des Systems erscheinen. Damit können Sie auf das BIOS zugreifen und z.B. die Bootreihenfolge ändern. Normalerweise führt das BIOS noch eine kurze Überprüfung Ihrer Hardware, beispielsweise der RAM-Chips, durch und begibt sich dann auf die Suche nach einem Bootloader. Es versucht, die ersten 512 Bytes von einem Diskettenlaufwerk oder einer Festplatte ins RAM zu laden und führt dann die dort enthaltenen Befehle aus.

Das BIOS hat also die Kontrolle an den ersten Teil des Betriebssystems übergeben: den Bootloader, ein Miniprogramm, das kaum ein halbes Kilobyte groß ist. In den 1980er Jahren genügte das völlig, um einen Kernel zu laden, doch moderne Bootloader müssen viele verschiedene Dateisysteme, Betriebssysteme und Grafikmodi unterstützen, weshalb 512 Bytes schon lange nicht mehr ausreichen.

Bei *Grub*, das die meisten Linux-Distributionen benutzen, lädt zum Beispiel ein kleines, nur ein halbes Kilobyte großes Bootloader-Programm sofort ein weiteres Programm, *Stage 1.5*. Dabei handelt es sich um einen etwas größeren Bootloader, der sich direkt vorne auf der Festplatte befindet, sodass er leicht gefunden werden kann. *Stage 1.5* lädt wiederum *Grub Stage 2*, einen voll entwickelten Bootloader mit allen bekannten Features. *Grub* liest eine Konfigurationsdatei, lädt den Linux-Kernel in den Arbeitsspeicher und startet ihn.

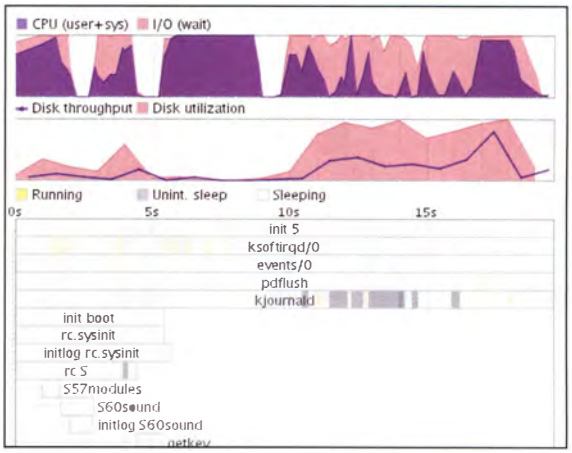
Linux-Kernel und Init

Wenn die ersten Bytes auf dem Linux-Kernel ausgeführt werden, dann ist er praktisch wie ein Neugeborenes, das noch keine Ahnung hat von der Welt seines Systems. Zunächst versucht er herauszufinden, welcher Prozessor und welche Anwendungen zur Verfügung stehen, er überprüft, wie viel Arbeitsspeicher installiert ist, und verschafft sich einen Überblick über das System. Dann weist er sich selbst einen sicheren Ort im Speicher zu – damit andere Systeme ihn nicht überschreiben und dadurch spektakuläre Abstürze verursachen können – und

beginnt schließlich, die Komponenten zu laden: Hardwaretreiber, Netzwerkprotokolle und so weiter.

Wenn der Kernel alle seine Aufgaben durchgeführt hat, ist es Zeit, die Kontrolle an das sogenannte Userland zu übergeben: den Ort, wo die Programme laufen. Der Kernel hat kein Interesse daran, *Bash*, *Gdm* oder ähnliche Programme selbst auszuführen, stattdessen startet er ein einziges Masterprogramm: **/sbin/init**. Es ist für den Start der Bootskripte zuständig, die den Computer zum Laufen bringen, aber es muss wissen, welche Skripte es starten soll. Die wichtigste Konfigurationsdatei für **init** ist **/etc/inittab**, eine einfache Textdatei, die Sie selbst verändern können. Diese Datei basiert auf einem Konzept, das Runlevel genannt wird – es bezeichnet die verschiedenen Betriebszustände, in denen das System sich befinden kann, wie etwa Einzelnutzer, Multiuser oder Shutdown. **/etc/inittab** gibt **init** den Befehl, das **/etc/init.d/rc**-Skript zu starten, wobei ein Parameter das Runlevel ist.

Das Skript ruft andere Skripte auf, die verschiedene Komponenten des Systems aufrufen: Die Netzwerkverbindung wird aufgebaut, die Systemprotokollierung initiiert und, falls Sie mit einem Desktop-Rechner arbeiten, das *X Window System* und der Login-Manager. Jetzt brauchen Sie nur noch Ihre Login-Daten einzugeben, und der Login-Manager startet Ihren Desktop und Ihren bevorzugten Window-Manager.



Sie möchten wissen, wie viel Zeit Ihre Bootskripte brauchen? Lassen Sie es sich in einer Grafik von *Bootchart* anzeigen (Sie finden *Bootchart* in den Paketsammlungen der meisten Distributionen).

Teil 2: Die Grub-Einstellungen bearbeiten

Wir wollen uns jetzt den Bootloader etwas genauer anschauen. Meistens wird hier *Grub* verwendet, ein leistungsstarkes Programm, das eine ganze Reihe von Betriebssystemen starten kann und Ihnen die Möglichkeit gibt, beim Hochfahren des Betriebssystems Veränderungen an der Konfiguration vorzunehmen. Um die Konfigurationsdateien und damit zusammenhängende Dienstprogramme kümmern wir uns später – jetzt geht es darum, wie Sie beim Hochfahren Veränderungen an unserer Installation von Debian vornehmen können.

Wenn *Grub* direkt nach dem BIOS-Bildschirm erscheint, erhalten Sie eine Liste von Boot-Optionen. Mit der Eingabetaste können Sie eine bestimmte Option starten, oder Sie können die Optionen sofort an Ort und Stelle bearbeiten. Wählen Sie den Eintrag, den Sie bearbeiten möchten, und drücken Sie E. Ein neuer Bildschirm wird aufgerufen mit drei Zeilen, die folgendermaßen aussehen:

```
root (hd0,0)
kernel /boot/vmlinuz-2.6.26-2-686 root=/dev/hda1 ro quiet
initrd /boot/initrd.img-2.6.26-2-686
```

Schauen Sie sich einmal die zweite Zeile an. Sie informiert *Grub* darüber, wo es den Linux-Kernel findet und welche Optionen an ihn weitergegeben werden sollen. Hier wird dem Kernel mitgeteilt, wo die Anwendung für die Root-Partition (/) steckt, dann befiehlt **ro**, dass die Partition nur schreibgeschützt gemountet werden soll. So können noch Überprüfungen der Dateisysteme durchgeführt werden – bis wenig später die Root-Partition beschreibbar neu eingehängt wird. Das **quiet** sagt dem Kern, dass er sich beim Ausspucken von neuen Nachrichten zurückhalten soll, wodurch der Bildschirm weniger zugemüllt wird und wir dem Hochfahrprozess leichter folgen können.

Diese Optionen können Sie bearbeiten, indem Sie mit den Pfeiltasten die darunterliegende Zeile wählen und wieder auf E drücken. Der Bildschirm geht daraufhin in den einfachen Bearbeitungsmodus, in dem Sie Optionen hinzufügen und entfernen können. Mit den Pfeiltasten bewegen Sie den Cursor in der Zeile. Versuchen wir es einmal: Fügen Sie ein einzelnes **s** nach **quiet** ein (mit einer Leerstelle dazwischen). Damit spezifizieren Sie den Runlevel, in dem der Kernel hochgefahren werden soll – **s** bedeutet *single user*, also Einzelnutzer.

Drücken Sie die Eingabetaste, um zu *Grub* zurückzukommen, dann drücken Sie B zum Starten des Bootprozesses. Weil wir das System im Einzelnutzermodus hochfahren, stoppt der normale Prozess, nachdem der Kernel initialisiert und die Root-Partition installiert wurde, und Sie werden nach dem Root-Passwort gefragt. Geben Sie es ein, dann landen Sie in einer Eingabeaufforderung. Diese erlaubt nur eingeschränkte Operationen, aber man kann damit gut Bootprobleme lösen – Sie können hier ungehindert Skripte bearbeiten und korrigieren. »

Booten Sie sich in die Zukunft

Klassischerweise wurden die Linux (und Unix) Bootskripte sequentiell durchgeführt – das heißt, ein Vorgang folgte auf den nächsten. Das ist einfach und garantiert, dass bestimmte Hardwarekomponenten und Anwendungen zum richtigen Zeitpunkt des Bootprozesses schon aktiviert sind. Allerdings ist es nicht gerade eine effiziente Methode und führt besonders bei älterer Hardware dazu, dass der Rechner ewig braucht, bis er hochgefahren ist. Denn über lange Zeiträume warten die Skripte nur darauf, dass etwas Bestimmtes geschieht: dass eine Hardwarekomponente sich selbständig aktiviert oder ein DHCP-Server auf dem Netzwerk eine Freigabe sendet, um zwei Beispiele zu nennen. Wäre es nicht super, wenn während dieser Warteperioden schon andere Prozesse laufen könnten? Genau das ist das Ziel von parallelisierten **init**-Skripten. Während also Ihr Netzwerk-Skript auf das DHCP wartet, entfernt ein anderes Skript schon **/tmp** oder startet *X Window System*. Doch um das zu erreichen, kann man nicht einfach das &-Zeichen ans Ende jedes Skriptbefehls hängen und alle parallel durchführen, denn einige Skripte laufen nur, wenn bestimmte Voraussetzungen ihnen zur Verfügung stehen. So muss z.B. ein Root-Skript, das eine IP-Adresse via DHCP erhält, davon ausgehen können, dass das Netzwerk schon von einem anderen Skript aktiviert wurde. *InitNG* ist ein parallelisiertes Bootsystem, in dem die Skripte zueinander in Abhängigkeiten stehen, die ihre Abfolge organisieren. *Upstart*, das von Ubuntu verwendet wird, startet Skripte aufgrund von Systemereignissen, also z.B. wenn eine Hardwarekomponente entdeckt wird. Dann gibt es noch *System D* (Fedora 15 wird damit arbeiten) und andere Ansätze. Man kann, allein schon den Dokumentierern und Administratoren zuliebe, nur hoffen, dass die Linux-Welt sich irgendwann auf ein System einigt. Doch in jedem Fall wird das Hochfahren von Linux-Rechnern durch der Trend zur Parallelisierung enorm beschleunigt.

Teil 3: Betrachten von Anmeldedateien

» Mit der oben erwähnten **quiet**-Option und allgemein der erhöhten Geschwindigkeit von modernen PCs ist es gut möglich, dass die Photonen der Bootnachrichten kaum Ihre Retina erreicht haben, und schon ist der Bootprozess abgeschlossen. Zum Glück können wir sie dann auch noch in Ruhe lesen, wenn das System hochgefahren ist und läuft. Schauen Sie in die Datei **/var/log/messages** (Sie müssen als Root angemeldet sein, damit Sie die Datei sehen können), und Sie sehen alles, was der Kernel generiert hat, exakt ab dem Moment, seit er seine Aufgaben durchführt. Weil der Kernel allerdings herausfinden möchte, in welcher Art von Hardware er existiert, wird er manchmal auch überrascht von dem, was er entdeckt. Deshalb keine Panik, falls so amüsante Warnmeldungen wie „warning: strange, CPU MTRRs all blank?“ auftauchen.

Quick-Tipp

Ein Hinweis zur Sicherheit: Jeder, der Zugriff auf Ihren Rechner hat, kann ihn rebooten und mit den *Grub*-Einstellungen herumspielen, auch wenn das Betriebssystem vollkommen geschützt ist. In einer späteren Lerneinheit geht es um die Konfigurationsdatei von *Grub*. Dort zeigen wir Ihnen, wie Sie den Bootloader durch ein Passwort schützen und solchen miesen Albernheiten den Riegel vorschieben können.

Saunders kommt zum Kern

Das Folgende beschreibt grob die Abfolge, nach der der Kernel vorgeht, wobei es allerdings einige Überschneidungen gibt:

» Holt sich Hardware-Info vom BIOS (diese muss nicht immer korrekt sein)

» Findet heraus, wie viele Prozessoren/Prozessorkerne es gibt, und verschafft sich einen Überblick über die CPU-Features

» Holt sich ACPI-Info und prüft den PCI-Bus auf Geräte

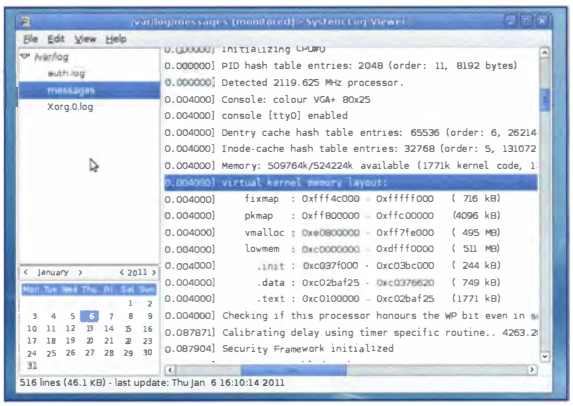
» Initiiert TCP/IP Netzwerkstack

» Sucht Festplatte, Disketten- und CD-ROM-Laufwerke

» Forscht nach USB-Controllern und angeschlossenen Geräten

Wenn der Kernel zufrieden ist mit dem Zustand des Systems, mountet er das Root-Dateisystem und führt wie oben beschrieben den Befehl **/sbin/init** durch.

/var/log/messages ist ein wertvolles Hilfsmittel, wenn man herausfinden möchte, was der Kernel schon seit dem



› **Eine etwas gefälligere Ansicht Ihrer Protokolleinträge erhalten Sie mit gnome-system-log oder *Kystemlog* von KDE.**

Hochfahren geleistet hat. Doch die Ausgabe kann auch ziemlich zugemüllt werden mit Informationen von anderen Programmen. In unserer Installation steht zum Beispiel in vielen Zeilen **debian**, doch es gibt auch andere, die **dhcbd** oder ähnliches enthalten.

Wenn Sie eine Liste ausschließlich mit den Aktivitäten des Kernels wollen, führen Sie den **dmesg**-Befehl durch (Sie können die Ausgabe zum bequemeren Lesen mit dem Zusatz **dmesg > listing.txt** in eine Textdatei umlenken). Die meisten Nachrichten werden aus der frühen Phase des Bootprozesses stammen, doch die Ausgabe wird aktualisiert, sobald Sie neue Hardware anschließen. Stecken Sie zum Beispiel einen USB-Stick ein, und führen Sie dann den Befehl durch. Neue Zeilen erscheinen, in denen beschrieben wird, wie der Kernel das Gerät entdeckt.

Teil 4: Runlevel und die Magie von /etc/init.d/

Die bereits erwähnten Runlevel (Betriebszustände) sind von enormer Bedeutung für Ihre Linux-Installation, denn sie definieren einen Zustand Ihres Systems und bestimmen damit, welche Prozesse ablaufen und welche Hilfsmittel zur Verfügung stehen. Es gibt sieben Runlevel, sechs davon werden mit Ziffern bezeichnet.

» **0** stoppt das System. Der Rechner geht in diesen Zustand, wenn er abgeschaltet wird. Ein Wechsel in dieses Runlevel löst Skripte aus, die die Prozessortätigkeit beenden und das System sauber zum Stillstand bringen.

» **1** Einzelnutzer-Betrieb. Normale Nutzer-Logins sind nicht erlaubt.

» **2 bis 5** Multiuser-Betrieb. In einer Debian-Installation sind diese Runlevel alle gleich. Bei Bedarf können Sie einen davon Ihren Bedürfnissen entsprechend anpassen. Dies ist der normale Betriebszustand, in dem mehrere Nutzer sich einloggen können und alle Komponenten aktiviert sind.

» **6** Reboot. Sehr ähnlich wie Runlevel 0.

Zuletzt gibt es noch das Runlevel S für den Einzelnutzerbetrieb, der Runlevel 1 ähnelt, aber auch einige Unterschiede beinhaltet: S ist das Runlevel, das beim Hochfahren verwendet wird, wenn sich der Rechner in einem sicheren Recovery-Modus befinden muss. Runlevel 1 benutzen Sie dagegen, wenn das System schon läuft und Sie in den Einzelnutzerbetrieb umschalten, etwa um Wartungsarbeiten durchzuführen.

Bei Debian sind die Runlevel 1-5 zwar identisch, doch in

Warnhinweise für andere Nutzer bei einem Runlevel-Wechsel

Auf einem Rechner, an dem nur ein Nutzer arbeitet, ist ein Wechsel des Runlevel kein Problem – da Sie selbst den Wechsel machen, sind Sie logischerweise darauf vorbereitet. Doch wie sieht es bei Rechnern aus, an denen mehrere Nutzer arbeiten? Was ist, wenn andere Nutzer sich über SSH eingeloggt haben und Programme auf dem Rechner laufen lassen? Die anderen Nutzer werden sicher nicht begeistert sein, wenn ihnen alles von einer Sekunde auf die nächste abstürzt. Zum Glück gibt es mehrere Möglichkeiten, wie Sie die anderen Nutzer über den Wechsel informieren können. Wenn Sie als Root angemeldet sind und **wall** eingeben, können Sie eine Nachricht schreiben und sie mit STRG+D beenden. Diese Nachricht erscheint dann auf dem Terminal von jedem zurzeit angemeldeten Nutzer. So können Sie z.B. den Warnhinweis „Shutdown in zehn Minuten“ durchgeben. Normale Nutzer können **wall** ebenfalls laufen lassen, doch sie können dabei den Empfang von Nachrichten von anderen normalen Nutzern mit dem **mesg**-Befehl verhindern. Alternativ können Sie die anderen Nutzer einfach anmailen, um sie zu warnen. Das ist nicht viel schwieriger und kann mit einem einzigen Befehl durchgeführt werden:

```
echo "Reboot in 10 minutes" | mail -s "Reboot notice" user@localhost
```

Wenn die Nutzer ein Email-Notification-Tool laufen lassen, erhalten sie die neue Nachricht sofort. Wenn Sie an einer großen Installation arbeiten, mit Hunderten von angemeldeten Nutzern, dann sollten Sie allerdings einige Stunden Vorwarnung geben und nicht nur ein paar Minuten.

einigen anderen Distributionen werden diese spezifiziert. So nutzen viele Distributionen Runlevel 3 im Multiuser-Betrieb für ein Anmelde-Setup im Textmodus und Runlevel 5 für ein grafisches Login. Mit dem Befehl **/sbin/runlevel** können Sie herausfinden, welches Runlevel Sie gerade benutzen. Um in ein anderes Runlevel zu wechseln, führen Sie als Root den Befehl **/sbin/telinit** durch, und zwar so:

```
/sbin/telinit 2
```

Wie finden Sie aber heraus, in welchem Runlevel Ihre Distribution standardmäßig läuft? Hier liegt die wahre Magie der **/etc/inittab**-Datei. Weiter oben in der Liste sehen Sie Zeilen wie diese:

```
# The default runlevel.
id:2:initdefault:
```

Zeilen, die mit dem #-Zeichen beginnen, sind Kommentare, während die nächste Zeile **init** mitteilt, dass Runlevel 2 der Standard sein soll. Falls Sie ein individuell angepasstes Runlevel einrichten, dabei die Skripte von Runlevel 3 verwenden und am liebsten immer direkt in dieses Runlevel hochfahren möchten, dann können Sie als Root einfach diese Datei ändern, die Zahl austauschen und Ihren Rechner neu starten. Ein bisschen weiter unten in der Datei **/etc/inittab** sehen Sie eine ganze Reihe von Zeilen wie die folgenden:

```
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
...
```

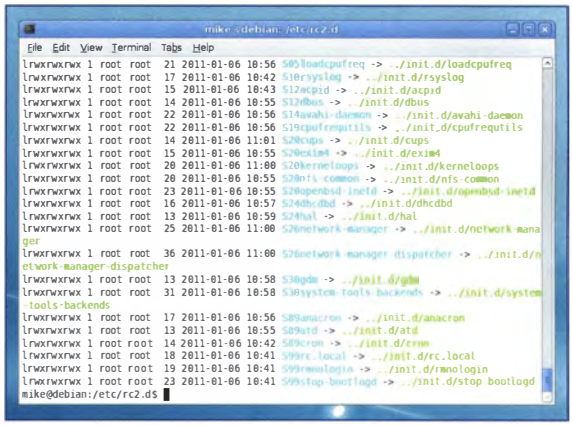
Das geht so weiter bis 6. Diese Zeilen sagen **init**, was es in jedem Runlevel tun soll, nämlich das Skript **/etc/init.d/rc** mit der Nummer des Runlevels als Parameter laufen zu lassen. Danach erkennt **/etc/init.d/rc** selbständig, welche Skripte es für das aktuelle Runlevel braucht.

Im Inneren eines Runlevels

Lassen Sie uns das Innenleben von **/etc/rc2.d** erforschen, des Standard-Runlevels in Debian. Sie finden hier Skripte wie **S05loadcpufreq** und **S89cron**, die jeweils für eine bestimmte Funktionalität in Ihrer Linux-Installation verantwortlich sind.

Da haben wir z.B. **S30gdm**, ein Skript, das den *Gnome Display Manager* startet. Aber was bedeuten die ersten drei Zeichen? **S** bedeutet, es ist ein Skript, das etwas startet, und **30** steht für die Position des Skripts in der Bootreihenfolge. Jedes Skript hat eine solche Nummer, und sie werden in der numerischen Reihenfolge nacheinander ausgeführt. Auf diese Weise ist garantiert, dass wichtige Skripte wie **S10rsyslog** am Anfang ausgeführt werden (um die Syslog-Protokollierung zu aktivieren), während nebensächlichere Anwendungen wie z.B. *Cron* (S89cron) erst gegen Ende des Runlevels ausgeführt werden.

Wenn Sie sich die Liste mit **ls -l** genauer anschauen, dann bemerken Sie, dass die Skripte keine wirklichen Dateien sind, sondern symbolische Links zu den Skripten in **/etc/init.d** – hier



› Jedes Runlevel hat ein Verzeichnis (/etc/rcX.d), in dem sich symbolische Links zu den Skripten in /etc/init.d befinden.

liegen die echten Skript-Dateien. Der Grund dafür ist, dass verschiedene Runlevel auf diese Skripte zugreifen. Vielleicht möchten Sie die Art und Weise verändern, wie *Cron* startet. Wenn Sie **/ect/init.d/cron** bearbeiten, dann werden Ihre Änderungen in allen Runlevels aktiv, die *Cron* benutzen.

Diese Skripte sind sorgfältig programmierte Wrapper, die die Programme umgeben, nach denen sie benannt sind. So ist z.B. **/etc/init.d/gdm** nicht nur eine Textdatei mit einer einzigen Zeile, die aus **gdm** besteht. Vielmehr werden darin die notwendigen Umgebungsvariablen angelegt, es werden Nachrichten zu den Protokolleinträgen hinzugefügt usw. In einem Debian-System können die meisten dieser Skripte mit bestimmten Parametern aufgerufen werden. Wenn Sie z.B. **/etc/init.d/gdm** eingeben, erhalten Sie eine Zeile wie diese:

```
Usage: /etc/ini.d/gdm {start|stop|restart|reload|forcereload|
status}
```

Sie können also **/etc/init.d/gdm start** ausführen und starten damit *Gdm*, mit **/etc/init.d/gdm halt** stoppen Sie das Programm. Beachten Sie, dass **restart** das Programm neu startet, während **reload** das Programm auffordert, seine Konfigurationsdateien neu einzulesen, ohne dass es – falls das möglich ist – gestoppt wird.

Zum Schluss möchten wir noch kurz über eine Sache bei **/etc/inittab** sprechen, die nichts mit den Runlevels zu tun hat, aber trotzdem sehr nützlich ist. Haben Sie sich je gefragt, woher eigentlich die Textterminals beim Hochfahren kommen? Die, zu denen Sie mit STRG+ALT+Fx vom *X server* wechseln können? Diese Terminals werden gegen Ende der **/etc/inittab**-Datei mit Zeilen wie den folgenden definiert:

```
2:23:respawn:/sbin/getty 38400 tty2
```

Der Teil **/sbin/getty 38400 tty2** ist einfach der Befehl, eine Login-Aufforderung auf dem zweiten, virtuellen Terminal durchzuführen. Sie können diesen Teil mit allem ersetzen, was Sie wollen – Sie könnten sogar ein virtuelles Terminal öffnen, auf dem *Tetris* läuft! **respawn** bedeutet, dass es nach jedem Abbruch immer wieder neu startet. ■

Wie schaltet man das System sicher ab?

In den Desktop-Betriebssystemen der 1980er Jahre gab es normalerweise kein besonderes Verfahren, um den Computer auszuschalten – man hat einfach auf den Schalter gedrückt, wenn man fertig war. Damals war das vollkommen in Ordnung, doch bei den heutigen Rechnern kann das sehr riskant sein, und zwar aus zweierlei Gründen.

Zum einen schreiben manche Betriebssysteme, auch Linux, die Daten nicht sofort auf die Festplatte, wenn sie eine Datei speichern. Sie warten, bis andere Abläufe ebenfalls Daten speichern wollen und bündeln dann alles zusammen in einer großen Speicheraktion, um die Rechnerleistung zu erhöhen. Allerdings können Sie Linux mit dem **sync**-Befehl dazu zwingen, alle Daten, die sich in seinen RAM-Puffern befinden, sofort auf die Festplatte zu speichern. Zum anderen gibt es zu den Linux-Startup-Skripten entsprechende Skripte für das Herunterfahren des Systems. Diese sorgen dafür, dass Betriebsabläufe sicher beendet werden, temporäre Dateien entfernt werden und so weiter. Zerbrechen Sie sich nicht den Kopf darüber, denn es ist keine Riesenkatastrophe, wenn diese Skripte nicht laufen. Doch sie helfen, damit Ihr System in einem ordentlichen Zustand bleibt. Die meisten von uns schalten den Rechner wahrscheinlich über ein Widget auf dem Desktop aus, doch wenn Sie das System lieber über die Befehlszeile herunterfahren wollen, dann lesen Sie im Handbuch nach, wie **shutdown**, **halt** und **reboot** funktionieren. Mit dem Befehl **shutdown** können Sie z.B. eine Verzögerung beim Abschalten angeben. Doch die gebräuchlichste Befehlszeile, um einen Rechner sofort auszuschalten ist **shutdown -h now**.

Lernen Sie Linux

MIT MIKE SAUNDERS

Lektion 3: In den ersten Lektionen ging es um die Hardware und den Boot-Prozess, jetzt kommen wir zum Aufbau des Dateisystems, zur Partitionierung und zu gemeinsam genutzten Programm-Bibliotheken.

Linux Professional Institute

Schätzen Sie mal, wie viele neue Dateien Sie nach der Installation einer einzigen CD mit Debian 6 auf Ihrer Festplatte haben. Die richtige Antwort ist: 82.698! Das hört sich fast unglaublich an – und unmöglich zu verwalten. Doch zum Glück erlaubt es der Aufbau des Linux-Dateisystems, dass diese unüberschaubare Menge von Dateien problemlos verarbeitet werden kann. Sie müssen nicht wissen, was die Aufgabe jeder dieser Dateien ist, doch an deren Speicherort im Dateisystem können Sie erkennen, wofür sie grob gebraucht wird und wo sie zum Einsatz kommt.

In dieser Lerneinheit sehen wir uns an, wie das Linux-Dateisystem aufgebaut ist, wie Sie Ihre Festplatte partitionieren und wie Sie die Konfiguration des Bootloaders *Grub* modifizieren.

Teil 1: Der Aufbau des Linux-Dateisystems

Wie ist eine Linux-Installation auf einer Festplatte organisiert? Anders als bei Windows, wo jedem Laufwerk und Zusatzgerät ein „Startpunkt" bzw. Laufwerksbuchstabe zugeteilt ist, gibt es bei Linux einen einzigen Ursprung – ein Art Daten-Urknall. Dieser Ursprung ist / oder das **root**-Verzeichnis, nicht zu verwechseln mit dem root-Nutzer (Administrator). Alle anderen Verzeichnisse sind dem root-Verzeichnis untergeordnet. Die folgenden Verzeichnisse und Links liegen im **root**-Verzeichnis:

» **/bin** Dieses Verzeichnis enthält wichtige Systemwerkzeuge und Dienstprogramme wie **ls**, **df**, **rm** – alles, was man braucht, um den Rechner hochzufahren und das System zu reparieren, sollte sich hier befinden.

» **/boot** Enthält die komprimierte Kernel-Image-Datei (**vmlinuz**) – das Programm, das vom Bootloader geladen und ausgeführt wird – sowie ein RAM-Disk-Abbild (**initrd**), das den Systemkernel mit einem minimalen Dateisystem und den nötigsten Treibern versorgt. Viele Distributionen legen noch die Datei **config** in dieses Verzeichnis – sie enthält die Einstellungen, um den Kernel aufzubauen –, und hier liegt auch ein **grub**-Unterverzeichnis für die Konfiguration des Bootloaders.

» **/dev** Hier werden die Geräteknoten abgelegt. In Linux können Sie auf die meisten Hardware-Komponenten zugreifen, als wären es Dateien, die gelesen und beschrieben werden können.

» **/etc** Hauptsächlich Konfigurationsdateien in einfachem Textformat. Die Boot-Skripte liegen ebenfalls hier. Alle Dateien hier sind systemübergreifende Konfigurationsdateien, z.B. für Programme wie *Apache*. Die Einstellungen für Desktop-Anwendungen findet man dagegen typischerweise im **home**-Verzeichnis des Nutzers.

» **/home** Hier sitzen normalerweise die **home**-Verzeichnisse. Jeder Nutzer hat eines.

» **initrd.img** Ein symbolischer Link (keine echte Datei, eher wie eine Verknüpfung bei Windows) zu der RAM-Disk-Datei im Verzeichnis **/boot**. Das vollständige Ziel des Links können Sie mit dem Befehl **ls -l** aufrufen.

» **/lib** Gemeinsame Bibliotheken. Wie **/bin** sind diese Bibliotheken zentral wichtig, um das System hochzufahren und grundlegend den Betrieb zu gewährleisten. Das Verzeichnis **/lib** enthält auch die Kernelmodule.

» **/lost+found** Falls Ihr Rechner während einer Schreiboperation abstürzen oder der Strom ausfallen sollte und er beim nächsten Booten die Festplatte überprüft (mit **fsck**), dann werden die Überreste der gegangenen Dateien hier gespeichert.

» **/media** Wenn externe Speichermedien wie USB-Sticks angeschlossen werden, wird ihnen beim automatischen Einhängen hier ein Verzeichnis zugewiesen.

» **/mnt** Ähnlich wie **/media**, nur dass dieses Verzeichnis normalerweise für manuell eingehängte Festspeicher verwendet wird, zum Beispiel für Festplatten oder Netzwerkfreigaben.

» **/opt** Optionale Software. Dieses Verzeichnis wird selten verwendet, doch in manchen Distributionen und Paketen können Sie hier große Programme finden.

» **/proc** Zugriff auf Prozessinformationen. Hier kann jeder Prozess (jedes laufende Programm), den das System ausführt, eingesehen werden.

» **/root** Hier liegen die persönlichen Dateien des Superusers bzw. des root-Accounts. Viele Administratoren bewahren hier auch Sicherungskopien der Konfigurationsdateien auf. Bei Multiuser-Installationen ist es lebenswichtig, dass normale Nutzer-Konten sich nicht in diesem Verzeichnis herumtreiben können.

» **/sbin** Hier liegen ausführbare Binärdateien, ähnlich wie in **/bin**, doch explizit nur für die Verwendung durch den Administrator. Das Verzeichnis enthält Programme, von denen normale Nutzer die Finger lassen sollten wie Dienstprogramme für Netzwerkkonfigurierung oder Partitionierung.

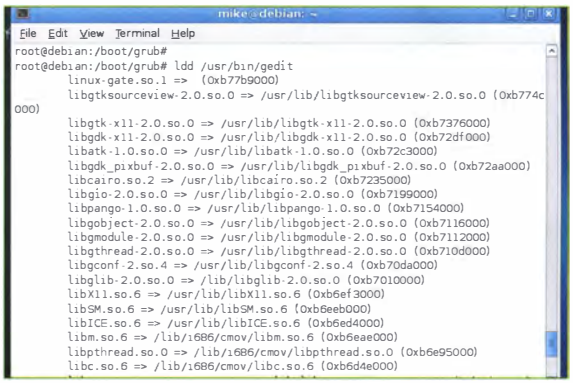
» **/selinux** Ein Platzhalter-Verzeichnis für Dateien, die von Security Enhanced Linux verwendet werden.

» **/srv** Dieses Verzeichnis ist für Daten bestimmt, die vom System verteilt werden. Die meisten Programme benutzen dafür allerdings **/var**.

» **/sys** Wie ein moderneres **/dev**-Verzeichnis mit zusätzlichen Fähigkeiten. Hier können Sie viele Informationen über die Hardware und den Kernel finden, aber für die normalen Administrationstätigkeiten ist das Verzeichnis nicht weiter wichtig.

» **/tmp** Verzeichnis für temporäre Dateien. Jedes Programm kann hier Daten ablegen, deshalb werden Sie hier allen möglichen Krimskram von Hintergrunddiensten, Web-Browsern usw. finden. Die meisten Distributionen leeren dieses Verzeichnis beim Hochfahren.

» **/usr** In diesem Verzeichnis sind wir in einer ganz anderen Welt. **/usr** enthält seine eigenen Versionen von **bin**-, **sbin** und **lib**-Verzeichnissen, doch diese gehören alle zu Anwendungen außerhalb des Primärsystems. Was notwendig ist, um den Rechner zum Laufen zu bringen, gehört in **/bin**, **/sbin** und **/lib**, während nicht unbedingt benötigte Programme wie *Firefox* und



› **Die meisten Programme wären längst nicht so funktional ohne den Zugriff auf die gemeinsamen Bibliotheken, wie der Befehl ldd zeigt.**

Emacs in diesem Verzeichnis ihren richtigen Platz finden. Dafür gibt es einen guten Grund: So können Sie das wichtige Basis-System auf einer Partition (/) haben und die Add-On-Programme auf einer anderen (**/usr**), wodurch Sie mehr Flexibilität erhalten. Auf diese Weise kann z.B. das Verzeichnis **/usr** über das Netzwerk eingehängt werden. Auf diesem Verzeichnis befindet sich auch **/usr/local**, das typischerweise für Programme verwendet wird, die Sie selbst kompiliert haben und die nicht beim Paket-Manager gespeichert werden sollen.

» **/var** Hier gehören Dateien hin, die „variabel" sind. Damit sind Dateien gemeint, die sich oft ändern, wie Protokolldateien, Datenbanken und E-Mail-Spools. Die meisten Distributionen legen auch *Apaches* Dokument-Basisverzeichnis (**/var/www**) hier ab. Auf stark ausgelasteten Servern, bei denen oft mit vielen Schreiboperationen auf dieses Verzeichnis zugegriffen wird, bekommt es häufig seine eigene Partition zugewiesen mit Modifikationen am Dateisystem für eine schnellere Performanz.

» **vmlinuz** Ein symbolischer Link zu der Kernel-Image-Datei in **/boot**.

Je nachdem, welche Distribution Sie benutzen, finden Sie vielleicht auch noch andere Dinge im **root**-Verzeichnis. Die meisten Distributionen halten sich jedoch an die Vorgaben des Filesystem Hierarchy Standard (FHS). Die Richtlinie ist ein Versuch, das Layout des Dateisystems für alle Distributionen zu standardisieren.

»

Was sind gemeinsame Bibliotheken?

Eine Bibliothek (englisch: *library*) ist ein Stück Code, das nicht allein für sich ausgeführt, aber von anderen Programmen genutzt werden kann. Angenommen, Sie schreiben einen Anwendung, die XML parsen muss, doch Sie möchten nicht zusätzlich einen ganzen XML-Parser programmieren. Dafür können Sie *libxml* einsetzen, eine Bibliothek, die XML analysiert und die ein anderer Programmierer bereits geschrieben hat. In einer normalen Linux-Installation gibt es Hunderte von solchen Bibliotheken, darunter auch welche für Standard-C-Funktionen (*libc*) und für Grafik-Schnittstellen (*libgtk*, *libqt*). Bibliotheken können statisch verlinkt sein – als Teil der ausführbaren Enddatei – doch normalerweise werden sie von allen Programmen nutzbar in **/lib**, **/usr/lib** und **/usr/local/lib** zur Verfügung gestellt und mit **.so** im Dateinamen bezeichnet, was für *shared object*, also geteiltes Objekt, steht. Das bedeutet, unterschiedliche Programme können auf dieselbe Bibliothek zugreifen, und wenn eine Sicherheitslücke entdeckt wird, reicht eine Korrektur, um alle Anwendungen abzudecken, die die Bibliothek nutzen. Gemeinsame Bibliotheken haben auch zur Folge, dass die Größe der Binärdateien des Programms kleiner ist und dadurch Speicherplatz gespart wird.

Mit **ldd** können Sie herausfinden, welches Programm welche Bibliotheken nutzt. So zeigt z.B. **ldd/usr/bin/gedit** eine Liste von Bibliotheken, darunter diese:

```
libgtk-x11-2.0.so.0 => /usr/lib/libgtk-x11-2.0.so.0
(0xb7476000)
```

Gedit hängt von *GTK* ab, weshalb *libgtk-x11* erforderlich ist. Aber wie wird der Pfad (Speicherort) bestimmt, an dem die Bibliotheken sitzen? Die Antwort steckt in **/etc/ld.so.conf**, die heutzutage auf alle Dateien in **/etc/ld.so.conf.d** zeigt. Diese Dateien enthalten simple Textzeilen mit den Speicherorten im Dateisystem, wo die Bibliotheken sich befinden, wie das Verzeichnis **/usr/local/lib**. Sie können hier neue Dateien mit Speicherorten hinzufügen, wenn Sie Bibliotheken irgendwo anders anlegen, doch danach müssen Sie **ldconfig** (als Root) durchführen, um den Cachespeicher zu aktualisieren, auf den der Programmlader zugreift. Es kann sein, dass Sie einmal ein Programm laufen lassen wollen, das eine Bibliothek an einem bestimmten Ort braucht, der aber nicht zu den üblichen Speicherorten für Bibliotheken gehört. Dafür können Sie die Umgebungs-Variable **LD_LIBRARY_PATH** verwenden. Wenn Sie z.B. die folgende Zeile eingeben, wird die **myprog** aus dem aktuellen Verzeichnis ausgeführt und **mylibs** vorübergehend auch zur Liste der Speicherorte für Bibliotheken hinzugefügt:

```
LD_LIBRARY_PATH=/path/to/mylibs ./
myprog.
```

Viele Computerspiele verwenden diese Methode, um Bibliotheken neben einer Binärdatei zu bündeln, ohne auf die Binärdateien zugreifen zu müssen.

Teil 2: Partitionsschemata

Ein Administrator muss selten eine Festplattenpartitionierung durchführen, doch kann diese Aufgabe schwerwiegende Folgen haben. Wenn einer Partition zu wenig oder zu viel Speicher zugewiesen wurde, stehen Sie später vor Riesenproblemen. Wie Sie die Festplatte partitionieren, hängt von dem Installer ab, der in Ihrer Distribution verwendet wird. Deshalb listen wir jetzt hier nicht Tausende Tastaturkürzel auf, sondern schauen uns ein Partitionierungswerkzeug genauer an, das alle Distributionen benutzen.

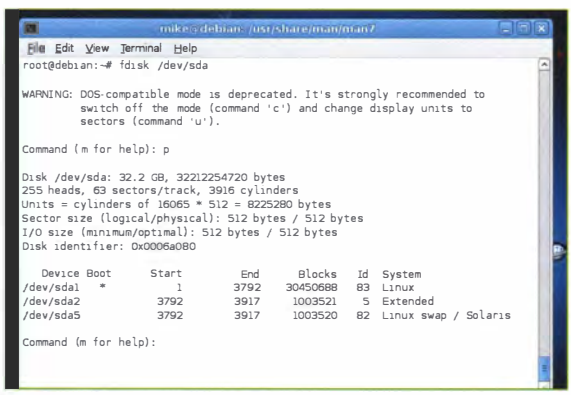
Öffnen Sie ein Terminal, wechseln Sie zu root und geben Sie ein:

```
fdisk /dev/sda
```

Ersetzen Sie **sda** durch der Gerätebezeichnung für Ihr Laufwerk (eine Festplatte: **sda**, Linux von zweitem Laufwerk: **sdb**, Überprüfung ggf. per **dmesg**). **fdisk** ist ein einfaches, befehlsgesteuertes Dienstprogramm für die Partitionierung. Wie *Vi* wirkt es sehr nüchtern und sachlich, aber es wird praktisch überall verwendet. Geben Sie **p** ein, und eine Liste der Partitionen auf der Festplatte erscheint. Tippen Sie **m**, und Sie erhalten die Liste der Ihnen zur Verfügung stehenden Befehle: Sie können Partitionen löschen (**d**), neue anlegen (**n**), Veränderungen auf der Festplatte speichern (**w**) usw.

Dies ist ein sehr leistungsfähiges Programm, aber es geht davon aus, dass Sie wissen, was Sie tun. **fdisk** formatiert die Partitionen nicht, es weist ihnen nur eine Typennummer zu. Um die Festplatte zu formatieren, geben Sie **mkfs** ein und drücken Sie auf Tab, was Ihnen die verschiedenen Möglichkeiten dieses Befehls auflistet. Wie Sie sehen, gibt es Befehle, um die Partition in typischen Linux-Formaten (wie **mkfs.ext4**), Windows FAT32 (**mkfs.vfat**) oder anderen Formaten zu formatieren.

Neben den Partitionen für das Dateisystem müssen Sie auch die Swap-Partition anlegen. Diese wird für den virtuellen Speicher verwendet. Anders ausgedrückt, wenn ein Programm nicht mehr in die RAM-Chips passt, weil andere Anwendungen den Arbeitsspeicher belegen, dann kann der Kernel dieses Programm auf die Swap-Partition schieben, indem er den Inhalt des vom Programm belegten Speichers dorthin schreibt. Wenn das Programm wieder aktiv wird, wird es von der Festplatte wieder in den Arbeitsspeicher geladen. Die meisten Administrationen empfehlen, dass die Swap-Partition mindestens doppelt so groß wie der Arbeitsspeicher sein soll, allerdings nicht größer als 2 GB. Sie können eine Partition mit **mkswap** als **swap** formatieren, darauf folgt die Gerätebezeichnung (z.B. **/dev/sda5**), und Sie aktivieren sie mit **swapon** plus Gerätebezeichnung.



› Die meisten Distributionen haben ihre eigenen grafischen Partitionierungswerkzeuge, doch egal, mit welcher Sie arbeiten, Sie werden immer das verlässliche fdisk vorfinden.

Partitionierungsmethoden

Welche Methode sollen Sie benutzen, wenn Sie eine Festplatte partitionieren? Es gibt allgemein drei Methoden:

1 All-in-one. Hier wird eine einzige große Partition angelegt, die die Dateien des Betriebssystems enthält, die Daten des **home**-Verzeichnisses, die temporären Dateien, die Server-Daten usw. In einigen Fällen ist dies nicht die effizienteste Lösung, aber es ist die bei weitem einfachste. Jedes Verzeichnis hat dabei das gleiche Anrecht auf Speicherplatz in der gesamten Partition. Viele desktoporientierte Distributionen verwenden standardmäßig diese Methode.

2 Aufteilung in **root** und **home**. Eine etwas komplexere Methode, bei der das **/home**-Verzeichnis in einer eigene Partition abgelegt wird, wodurch es getrennt von der **root**(/)-Partition ist. Der große Vorteil dieser Methode ist, dass Sie Distributionen upgraden, neu installieren und verändern können, während die persönlichen Daten und Einstellungen in **/home** intakt bleiben. Diese Aufteilung kann auch hilfreich sein, wenn den beiden Partitionen der Speicherplatz so zugewiesen wird, das eine überquellende **/home**-Partition nicht das System lahmlegt.

3 Getrennte Partitionen für alles. Wenn Sie an einem Zentralrechner arbeiten, wie z.B. einem mit dem Internet verbundenen Server, der rund um die Uhr laufen muss, dann können Sie einige sehr nützliche Partitionssysteme entwickeln. Nehmen wir an, Ihr Rechner hat zwei Festplatten; die eine läuft langsam, die andere schnell. Wenn Sie einen ziemlich gut ausgelasteten E-Mail-Server betreiben, dann können Sie das **/root**-Verzeichnis auf der langsamen Festplatte anlegen, weil es sowieso nur für das Hochfahren und das Laden von ein paar wenigen Programmen benutzt wird. Auf der schnellere Festplatte dagegen sollte **/var/spool** angelegt werden, weil dort Hunderte von Lese-und Schreiboperationen pro Minutenstattfinden können.

Diese Flexibilität der Partitionierung ist eine der großen Stärken von Linux und Unix, und die Vielfalt der Anwendungsmöglichkeiten nimmt immer weiter zu. Nehmen Sie als Beispiel die neuen, schnellen SSD-Laufwerke: Sie können **/home** auf eine herkömmliche Festplatte legen und haben damit für wenig Geld sehr viel Speicherplatz gewonnen. Dass **/root**-Verzeichnis legen Sie aber auf das SSD-Laufwerk, und Ihr System bootet und lädt Programme praktisch in Lichtgeschwindigkeit.

Quick-Tipp

Sie möchten nicht, dass andere Nutzer mit den Optionen des *Grub* Boot-Bildschirms herumspielen? Dann schützen Sie Ihre Boot-Einträge durch ein Passwort, so dass nur Sie sie bearbeiten können – eine Anleitung hierzu finden Sie in der online-Dokumentation von *Grub* unter **http://tinyurl.com/6czhkn8**.

Testen Sie Ihr Wissen!

Hier ein paar Aufgaben zu den drei Teilen in dieser Lektion:

1 Wo befinden sich das Kernel-Image und die RAM-Disk-Dateien?

2 Erläutern Sie den Unterschied zwischen **/lib**, **/usr/lib** und **/usr/local/lib**.

3 Erläutern Sie die möglichen Methoden, mit denen in Linux die Festplatte partitioniert werden kann. Warum kann es sinnvoll sein, **/home** auf eine gesonderte Partition zu legen?

4 Beschreiben Sie, wie man im System einen neuen Speicherort für eine Bibliothek hinzufügt.

Falls Sie Schwierigkeiten haben, gehen Sie einfach noch einmal zurück und lesen Sie es sich in Ruhe durch. Am besten verinnerlichen Sie die erlernten Informationen, wenn Sie auf Ihrem Rechner damit experimentieren (oder mit einer Distribution auf *VirtualBox*, falls Ihnen das Risiko zu groß ist, dass Sie Ihre Installation ruinieren).

Teil 3: Wie konfiguriert man den Bootloader?

Heute verwenden fast alle Linux-Distributionen *Grub2* (der *Grand Unified Bootloader*, wie er mit vollem Namen heißt), um den Kernel zu laden und den gesamten Linux-Bootprozess zu starten. In der vorigen Lerneinheit haben wir *Grub* unter die Lupe genommen und uns besonders angeschaut, wie man die Optionen in *Grub* selbst bearbeiten kann. Aber Ihre Veränderungen sind nur temporär. Wenn es um etwas geht, das Sie jedes Mal aufs Neue einstellen und Sie so bei jedem Hochfahren die Bootsequenz unterbrechen müssen, dann kann das ganz schön nervig werden. Das Problem können Sie lösen, indem Sie die **/etc/default/grub**-Datei ändern.

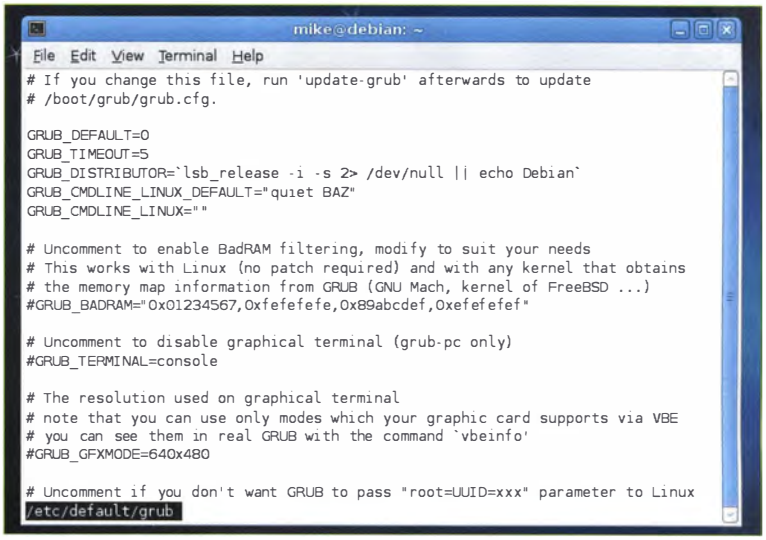
Diese Datei ist nicht die Konfigurationsdatei von *Grub* selbst – diese liegt im Verzeichnis **/boot/grub/grub.cfg**. Doch die Konfigurationsdatei wird automatisch von Skripten angelegt, wenn der Kernel ein Update durchführt, deshalb sollten Sie sie eigentlich nie manuell ändern müssen. In den meisten Fällen möchten Sie wahrscheinlich eine Option zum Boot-Befehl des Kernels hinzufügen, z.B. um ein Zubehörteil zu deaktivieren oder um das System in einem bestimmten Modus hochzufahren. Sie fügen solche Optionen hinzu, indem Sie als Root **/etc/default/grub** in einem Text-Editor öffnen und sich diese Zeile anschauen:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
```

Hier sehen Sie die Standard-Optionen, die an den Linux-Kernel weitergegeben werden. Fügen Sie die Optionen, die Sie brauchen, nach **quiet** ein, getrennt durch eine Leerstelle und innerhalb der Anführungszeichen. Wenn Sie alles eingegeben haben, führen Sie diesen Befehl durch:

```
/usr/sbin/update-grub
```

Dadurch wird ein Update von **/boot/grub/grub.cfg** mit den neuen Optionen gemacht.



› Der Ort, an dem Sie Boot-Optionen für den Kernel hinzufügen können, ist /etc/default/grub. Vergessen Sie nicht, danach update-grub durchzuführen, damit Ihre Modifizierungen auch in die Datei /boot/grub/grub.cfg übertragen werden.

Der alte Grub

Falls Sie eine ältere Distribution mit *Grub1* verwenden, sieht der Aufbau etwas anders aus. Rufen Sie die Datei **/boot/grub/menu.lst** auf, in der Sie ähnliche Einträge wie die folgenden finden:

```
title Fedora Core (2.6.20-1.2952.fc6)
root (hd0,0)
kernel /vmlinuz-2.6.20-1.2952.fc6 ro root=/dev/md2 rhgb
quiet
initrd /initrd-2.6.20-1-2952.fc6.img
```

Hier können Sie Optionen direkt ans Ende der **kernel**-Zeile anfügen. Speichern Sie und rebooten Sie, damit die Optionen aktiv werden.

Sollte *Grub* beschädigt oder von einem anderen Bootloader entfernt werden, können Sie es wieder installieren, indem als Root den folgenden Befehl ausführen.

```
grub-install /dev/sda
```

Ersetzen Sie **sda** durch **sdb**, falls Sie das Programm auf Ihrer zweiten Festplatte installieren möchten. Dadurch wird der Anfangsteil von *Grub* auf Ihre Festplatte geschrieben, und zwar auf die ersten 512 Bytes, die auch Master Boot Record (MBR) genannt werden. *Grub* muss nicht immer auf dem MBR installiert werden, es kann auch in den Superblock (den ersten Sektor) einer Partition geschrieben werden. So kann dann ein übergeordneter Bootloader auf dem MBR eine ganze Reihe von Bootloadern nachladen. Das wollen wir hier aber nicht weiter vertiefen.

Grub wird zwar von der großen Mehrheit der Distributionen verwendet, doch es sind immer noch ein paar im Umlauf, die den älteren LILO verwenden – den *Linux Loader*. Seine Konfigurationsdatei ist **/etc/lilo.cfg**, und wenn Sie dort irgendwelche Veränderungen gemacht haben, führen Sie **/sbin/lilo** durch, damit die im Boot-Sektor gespeicherten Einstellungen aktualisiert werden. ■

Quick-Tipp

Ein Einhängepunkt ist der Ort, an dem eine Partition mit dem Dateisystem verbunden ist. Nehmen wir einmal an, Sie nutzen die **/dev/sdb2**-Partition auf Ihrer Festplatte für die **home**-Verzeichnisse, dann ist der Einhängepunkt **/home**. Ganz einfach!

Die Magie von /etc/fstab

Nachdem Sie nun über die verschiedenen Partitionierungsmethoden Bescheid wissen und erfahren haben, wie Sie sie mit **fdisk** und **mkfs** anlegen, fragen Sie sich vielleicht, wie die Partitionen in eine laufende Linux-Installation eingebunden sind. Kontrolliert wird das von **/etc/fstab**, einer einfachen Textdatei, die Partitionen mit Einhängepunkten verbindet. Schauen Sie in die Datei hinein, und Sie finden etliche Zeile, die so oder ähnlich aussehen:

```
UUID=cb300f2c-6baf-4d3e-85d2-9c965f6327a0 /
ext3 errors=remount-ro 0 1
```

Diese Zeile unterteilt sich in fünf Bereiche. Der erste bezieht sich auf das Gerät, das Sie in **/dev/sda1-type format** oder mit einer modernen UUID-Sequenz genauer spezifizieren können (führen Sie den **blkid**-Befehl mit einer Gerätebezeichnung aus, dann erhalten Sie die UUID – dies ist nur eine besondere Art der Identifizierung eines bestimmten Geräts). Dann folgt der Einhängepunkt, der in diesem Beispiel im **root**-Verzeichnis liegt. Es folgt der Typ des Dateisystems, und schließlich können noch Optionen hinzugefügt werden. Hier sagen wir, dass, falls es zu Fehlern kommt, wenn die Boot-Skripte die Festplatte einhängen, diese neu eingehängt werden soll, allerdings nur im schreibgeschützten Modus, damit nicht durch Schreiboperationen noch mehr Schaden angerichtet wird. Schauen Sie auf der entsprechenden Man-Page nach, welche Optionen für jedes Dateisystem zur Verfügung stehen. Die Zahlen ganz am Ende haben mit den Überprüfungen des Dateisystems zu tun. Diese Standardeinstellungen brauchen Sie nicht zu verändern.

Sie können Ihre eigenen Einhängepunkte mit einem Text-Editor zu **/etc/fstab** hinzufügen. Beachten Sie, dass manche Distributionen automatisch Veränderungen in der Datei vornehmen, deshalb unser Tipp, dass Sie auf jeden Fall eine Sicherungskopie der Originaldatei machen sollten.

Lernen Sie Linux

MIT MIKE SAUNDERS

Lektion 4: Ein Thema, mit dem sich alle Administratoren gut auskennen müssen, ist das Paketmanagement. Lernen Sie hier, wie Paketmanagement mit Debian und RPM funktioniert.

Linux Professional Institute

Software auf Linux installieren – kinderleicht, oder? Sie fahren Ihren Grafik-Browser hoch, klicken die Kästchen bei den gewünschten Anwendungen an, und dann werden diese heruntergeladen und installiert. So funktioniert das für die meisten Nutzer. Wenn Sie allerdings ein kompetenter Systemadministrator werden möchten, dann müssen Sie auch dem Grunsatz nach verstehen, wie Paketmanagement funktioniert.

Für (relative) Beulinge in der Linux-Welt wollen wir uns zuerst anschauen, was eigentlich ein Paket ist. Das ist kurz gesagt eine einzige komprimierte Datei, aus der sich viele Dateien und Verzeichnisse extrahieren lassen. Viele Pakete enthalten Programme, aber sie können auch Bildmaterial und Dokumentation enthalten. Große Projekte werden von den Erstellern von Distributionen in etliche Pakete aufgeteilt. Wenn dann für ein kleines Programm ein Sicherheitsupdate ansteht, müssen Sie nicht den gesamten Desktop herunterladen.

Normalerweise sind Pakete allerdings komplexer als einfache Archive. Zum Beispiel können sie abhängig von anderen Paketen sein oder Skripte enthalten, die man bei der Installation und Deinstallation ausführen sollte.

Teil 1: So funktioniert es mit Debian

Beginnen wir mit den Debian-Paketen. Sie wurden ursprünglich von dem Debian-Projekt entwickelt und werden inzwischen in einem Großteil der auf Debian basierenden Distributionen verwendet, wie z.B. in Ubuntu.

So sieht der Dateiname eines typischen Debian-Pakets aus:

```
nano_2.2.4-1_i386.deb
```

Er besteht aus fünf Elementen. Diese sind: der Name der Anwendung (**nano**), die Versionsnummer (**2.2.4**), die distributionseigene Version des Pakets (**-1**), die zugehörige CPU-Architektur (**i386**) sowie die Endung **.deb**, welche die Datei als ein Debian-Paket ausweist.

Nehmen wir an, Ihr Rechner läuft mit Debian 6, Sie haben dieses **Nano**-Paket aus dem Netz heruntergeladen, und nun sitzt es in Ihrem **home**-Verzeichnis. Navigieren Sie zu Anwendungen > Zubehör > Terminal und geben Sie **su** ein, um in die Rolle des Superusers (also des Administrators) zu wechseln. Das Paket installieren Sie mit:

```
dpkg -i nano_2.2.4-1_i386.deb
```

Nach der Installation können Sie das Kommando **nano** eingeben, um das Programm zu starten. *Dpkg* ist ein nützliches Dienstprogramm, mit dem man ein oder auch mehrere Pakete installieren kann, die schon aus dem Internet heruntergeladen wurden. Wenn Sie viele Pakete installieren müssen, benutzen Sie **dpkg -i *.deb/** (das Sternchen dient als Platzhalter für alle Dateien, die mit **.deb** enden).

Es gibt zwei Möglichkeiten, wie Sie ein Paket wieder entfernen können. Wenn Sie **dpkg -r nano** eingeben, wird das Programm zwar gelöscht, doch die Konfigurationsdateien bleiben intakt. Möchten Sie das Paket jedoch vollständig löschen, führen Sie den Befehl **dpkg –purge nano** aus.

Das alles funktioniert gut, wenn Sie schon heruntergeladene Pakete installieren, eine flexiblere Methode ist jedoch **apt-get**.

APT ist die Abkürzung von *Advanced Package Tool* (ein Paketverwaltungssystem), das Möglichkeiten bietet, die über das einfache Installieren und Entfernen von Paketen hinausgehen. Vor allem kann **apt-get** Pakete (und Abhängigkeiten) aus dem Internet abrufen. Stellen Sie sich vor, Sie hätten gerne den *Vim*-Editor, aber Sie haben nicht die entsprechenden Deb-Pakete installiert. Nun geben Sie ein:

```
apt-get install vim
```

APT sucht Ihnen im Internet die richtigen Pakete für Ihre aktuelle Distributions-Version zusammen und installiert sie. Direkt vor dem Downloaden bekommen Sie noch die Möglichkeit, die Operation zu bestätigen:

```
Need to get 7,005 kB of archives.
After this operation, 27.6MB of additional disk space will be
 used.
Do you want to continue [Y/N]?
```

Hier wird Ihnen mitgeteilt, welche Auswirkung die Installation auf Ihren Speicherplatz haben wird (die beiden Zahlen beziehen sich einmal auf den Download-Umfang und dann auf die Größe im entpacktem Zustand). Drücken Sie **Y**, um fortzufahren.

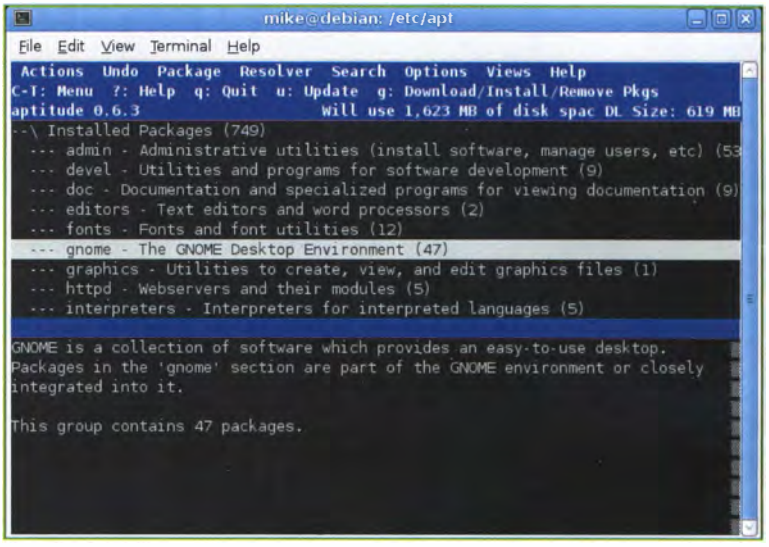
Woher weiß APT nun, wo es die Pakete findet? Die Antwort heißt: Repositories. Dies sind strukturierte Online-Sammlungen von Paketen für eine bestimmte Version einer Linux-Distribution. Die Pakete wurden schon auf ihre Funktionsfähigkeit und ihre Eignung für die jeweilige Distribution hin überprüft, und alle etwaigen Abhängigkeiten, die das Paket braucht, wurden hinzugefügt. Repositories können riesige Archive mit Tausenden von Paketen sein, aber auch kleine Privatsammlungen, die sich irgendwo im Hinterland des Internets versteckt halten.

Weil Repositories online stehen, können sie mit URLs bezeichnet werden. Rufen Sie einmal **/etc/apt/sources.list** auf, und Sie erhalten Zeilen wie die folgende:

```
deb http://ftp.uk.debian.org/debian/ squeeze main
```

Hier teilt **deb** dem APT-Paketmanager mit, dass es sich bei der URL um eine Quelle von Debian-Paketen handelt. Darauf folgt die URL selbst. Danach sehen Sie die Versionsbezeichnung der Distribution, die in diesem Beispiel **squeeze** heißt, womit Debian 6 gemeint ist. Ganz am Ende steht die Art der Pakete, die Sie aufrufen wollen. In Debian gibt es z.B. eine Hauptkategorie (main category) für Pakete, die sich nach den Debian-Richtlinien für Freie Software richten. Aber es gibt auch eine eingeschränkte Kategorie für Software, die nicht so offen, also proprietär ist.

In der Hauptkategorie finden Sie also allgemein Programme, die Sie vielleicht installieren möchten. Aber es gibt auch ein Repository für Sicherheits- und Bugfix-Aktualisierungen, das Sie hier finden:



› Nur weil Sie mit der Befehlszeile arbeiten, brauchen Sie noch lange nicht auf einen guten Paketmanager zu verzichten – *Aptitude* ist da genau der Richtige für Sie.

```
deb http://security.debian.org/ squeeze/updates main
```

Zunehmend bieten Linux-Softwarehersteller auch ihre eigenen Repositories an, zusätzlich zu den offiziellen der Distribution. Wenn Sie für die Paketinstallation eine Textzeile ähnlich der obigen erhalten haben, kopieren Sie Zeile in Ihre **/etc/apt/sources.list**-Datei und speichern Sie sie. APT legt einen lokalen Zwischenspeicher mit Paketinformationen an, die ein schnelleres Suchen erlauben, doch das Programm muss erst ein Update durchführen, damit auch die Informationen zu den neuen Paketen dort gespeichert werden. Ein Update erreichen Sie durch folgenden Befehl:

```
apt-get update
```

Danach können Sie die neuen Pakete installieren (um alle aktualisierte Pakete auf einmal zu installieren, benutzen Sie **apt-get upgrade**).

APT ist ein sehr leistungsstarkes System, das mit einigen Dienstprogrammen eingesetzt werden kann (geben Sie **apt** ein und drücken Sie auf TAB, dann erscheinen die zur Auswahl stehenden Optionen). Einen Großteil der Funktionalität von APT können Sie in einem einzigen Programm bündeln, indem Sie **aptitude** eingeben. Dabei handelt es sich um ein auf *Ncurses* basierendes Programm, das ein rudimentäres GUI für den Textmodus besitzt, mit Menüs, Dialogboxen usw. Sogar ein *Minesweeper*-Clone ist mit eingebaut! »

Quick-Tipp

Wenn Sie Programme mit **apt-get install** installiert haben, werden die heruntergeladenen Pakete in einem Zwischenspeicher unter **/var/cache/apt/archives** abgelegt, damit sie später wiederverwendet werden können. Wenn Sie Monsterprogramme wie KDE installieren, kann dieser Zwischenspeicher allerdings ziemlich groß werden. Um ihn aufzuräumen und Platz zu schaffen, führen Sie den Befehl **apt-get clean** durch.

Paketkonvertierung mit Alien

In der Linux-Welt sind RPM und Deb die beiden bekanntesten Paketformate, doch sie arbeiten nicht gut zusammen. Natürlich können Sie die *Dpkg*-Tools auf einen RPM-Rechner installieren (oder den **rpm**-Befehl auf einem Debian-Rechner ausführen) und versuchen, die Pakete so zwangsweise zu installieren. Doch das Ergebnis wird nicht schön sein, und Sie müssen mit etlichen Schäden rechnen. Die vernünftigere Lösung ist da das *Alien*-Programm, das Sie in den Debian-Repositories finden. Dieses nützliche kleine Dienstprogramm konvertiert Deb-Dateien in RPMs und umgekehrt. Zum Beispiel:

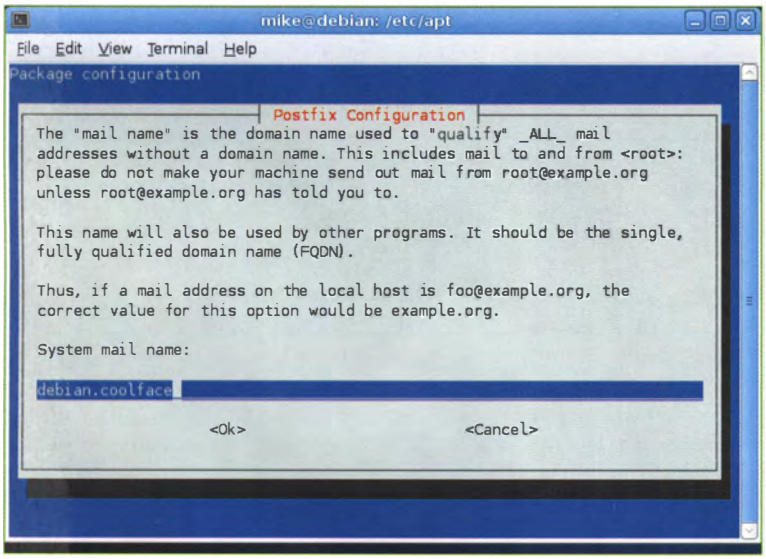
```
alien --to-deb nasm-2.07-1.i386.rpm
```

Der Befehl erzeugt eine Datei mit der Bezeichnung **nasm_2.07-2_i386.deb**, die Sie dann mit dem oben beschriebenen Befehl **dpkg -i** installieren können. Dass die Installation dann auch funktioniert, ist damit allerdings noch nicht sicher. Manche Pakete sind so spezifisch auf eine bestimmte Distribution zugeschnitten, dass sie einfach auseinanderfallen, wenn man versucht, sie mit einer anderen Distribution zu verwenden. *Alien* verändert nur das Kompressionsformat und die Metadaten-Formate, damit sie zu einem bestimmten Paketformat passen. Es kann nicht garantieren, dass das Paket auch die Richtlinien der Distribution für das Dateisystem-Layout beachtet, oder dass Vor- und Nachinstallations-Skripte richtig funktionieren.

Sie können die Paketlisten mit der Eingabe- und den Corsortasten durchsuchen, die zur Verfügung stehenden Tastaturbefehle werden am oberen Rand angezeigt. Wenn Sie STRG+T drücken, erscheint ein Menü. *Aptitude* ist besonders hilfreich, wenn Sie über SSH in einen Remote-Computer eingeloggt sind und eine bestimmte Aufgabe ausführen müssen, sich aber nicht mehr an den genauen Befehl erinnern. In *Aptitude* können Sie einfach im Menü nachschlagen, anstatt kompliziert nach dem Befehl zu suchen.

Gehen wir für einen Moment zurück zu dem *Dpkg*-Dienstprogramm. Außer dem Installieren und Entfernen von Paketen kann *Dpkg* auch eine Liste der schon in der Datenbank installierten Pakete abrufen. Wenn Sie z.B. wissen möchten, welche Dateien sich im **nano**-Paket befinden, geben Sie diesen Befehl ein:

```
dpkg -L nano
```



Sie möchten die Einstellungen eines Programms mittels seines Paket-Skripts ändern? Mit dem dpkg-reconfigure-Befehl ist das kein Problem.

Und um eine noch detaillierte Liste mit Informationen über ein Paket zu erhalten, führen Sie diesen Befehl durch:

```
dpkg -s nano
```

Damit erfahren Sie alles, was Sie über ein Paket wissen müssen: Version, Größe, Architektur, Abhängigkeiten und sogar die Email-Adresse des Paketbetreuers, falls Sie ihm ein Problem melden möchten (wobei es in diesem Fall oft besser ist, wenn Sie die Bugtracking-Software der Distribution verwenden). Eine interessante Information bietet hier die **provides**-Zeile. *Nano* z.B. bietet die Editier-Funktion, die mit Absicht sehr allgemein gehalten ist. Einige andere Befehlszeilen-Tools sind auf einen installierten Text-Editor angewiesen, aber es wäre etwas witzlos, wenn sie einen ganz bestimmten Editor, wie *Emacs* oder *Vim*, spezifizieren würden. Stattdessen verlangen sie, dass ein Paket mit einem Editor installiert wird – und *Nano* bietet genau das.

Ein weiterer nützlicher Befehl ist **dpkg -S**, gefolgt von einem Dateinamen. Er sucht nach Dateien im System, die diesem Dateinamen entsprechen, und teilt Ihnen dann mit, in welchem Paket sich die Dateien befinden. So lokalisiert z.B. **dpkg -S vmlinuz** die **vmlinuz**-Kerneldatei im System und zeigt Ihnen an, welches Paket ursprünglich die Installation ausgeführt hat.

Zum Schluss noch ein Wort über Paketkonfigurationen. Wie Sie wissen, befinden sich bei vielen Programme textbasierte Konfigurationsdateien im **/etc**-Verzeichnis, die Sie manuell verändern können. Das ist schon hilfreich, aber viele Debian-Pakete möchten es dem Administrator noch einfacher machen, indem sie in einem gewissen Umfang automatisierte Abläufe bieten.

Installieren Sie den *Postfix*-Mail-Server, z.B. mit **apt-get**, und es öffnet sich ein Dialogfenster, mit dem Sie durch den Setup-Prozess für den Server geleitet werden. So brauchen Sie sich gar nicht erst nicht mit dem Format dieser speziellen Konfigurationsdatei auseinanderzusetzen. Und falls Sie die Konfiguration doch einmal ändern müssen und es auf die herkömmliche Debian-Weise machen möchten, dann verwenden Sie einfach diesen Befehl:

```
dpkg-reconfigure postfix
```

Pakete über den Quellcode selbst erstellen

Wahrscheinlich sind die meisten Binärpakete aus einem entsprechenden Quellcode entwickelt worden. Paketentwicklung bedeutet aber nicht, dass einfach eine Binärdatei gezippt wird – der Prozess ist um Einiges komplizierter, man benötigt Skripte und Konfigurationsdateien dazu. Der erste Schritt in auf Debian basierenden Distributionen ist die Installation der nötigen Tools. Das machen Sie mit diesem Befehl:
apt-get install dpkg-dev build-essential fakeroot
Danach teilen Sie Debian mit, dass Sie Zugriff auf den Quellcode möchten und nicht nur auf die binären Debs. Dazu öffnen Sie **/etc/apt/sources.list** und duplizieren die Zeilen, die mit **deb** bis **deb-src** beginnen, z.B.:

```
deb-src http://ftp.uk.debian.org/debian/
squeeze main
```

Den Quellcode für ein Programm können Sie sich mit **apt-get source package** holen, wobei Sie **package** durch den Namen des Programms ersetzen. Der ursprünglich hochgeladene Quellcode wird heruntergeladen und extrahiert, und Veränderungen werden vorgenommen, die spezifisch für Ihre Distribution sind. In das bei der Extraktion entstandene Verzeichnis wechseln Sie mit **cd package-***. Einige Pakete haben Bibliotheks- und andere Dienstprogramme mit Abhängigkeiten, die erst bei der Paketentwicklung zum Tragen kommen; diese können Sie mit **apt-get build-dep packages** installieren. Jetzt können Sie alle Modifizierungen am Quellcode vornehmen, die Sie brauchen, oder auch die Optimierungs-Möglichkeiten für den Compiler in der **CFLAGS**-Zeile in **debian/rules** ändern. Dann erstellen Sie das Paket mit diesem Befehl:

```
dpkg-buildpackage
```

Sobald der Prozess abgeschlossen ist, geben Sie **cd ..** ein, um in das über dem aktuellen liegende Verzeichnis zu gelangen, dann geben Sie **ls** ein. Sie finden dort eines oder mehrere neu entwickelte Deb-Pakete vor, die Sie nun auch weitergeben können.
Wenn Sie mit einem RPM-System arbeiten, können Sie *Yumdownloader* installieren, mit dem Sie sich SRPM (S steht für *source*, also RPM-Quellcode-Pakete) holen können, und zwar mit **yumdownloader -source package** (wobei Sie **package** mit dem ersetzen, was Sie gerade brauchen). Ein SRPM enthält zum einen den Quellcode, zum anderen die Angaben, um den Code zu bauen (eine SPEC-Datei) und ggf. distributionsspezifische Patches und Aktualisierungen.
Mit dem Befehl **rpmbuild –rebuild filename.src.rpm** können Sie dann aus den SRPM Binärpakete bauen. Je nachdem, was für ein Programm Sie entwickeln, erhalten Sie am Ende ein oder mehrere binäre RPM-Pakete, die Sie verteilen und installieren können.

Teil 2: So funktioniert es mit RPM

Ursprünglich war dieses Paketverwaltungssystem als der *Red Hat Package Manager* bekannt, doch inzwischen wird die ehemalige Abkürzung als Name verwendet, um zu betonen, dass der *RPM Package Manager* nicht auf eine Distribution festgelegt ist. Viele Distributionen benutzen *RPM*, bei dieser Lerneinheit arbeiten wir mit CentOS 5, einer extrem zuverlässigen, auf Red Hat Linux aufbauenden Distribution.

Das normale Paketmanagement auf einem *RPM*-System wird, wie Sie sicher schon geahnt haben, mit dem **rpm**-Befehl durchgeführt. Damit können Sie mit den Paketen arbeiten, nachdem Sie sie heruntergeladen haben. Nehmen wir an, Sie holen sich ein Paket für *NASM*:

```
rpm -Uvh nasm-0.98.39-1.i396.rpm
```

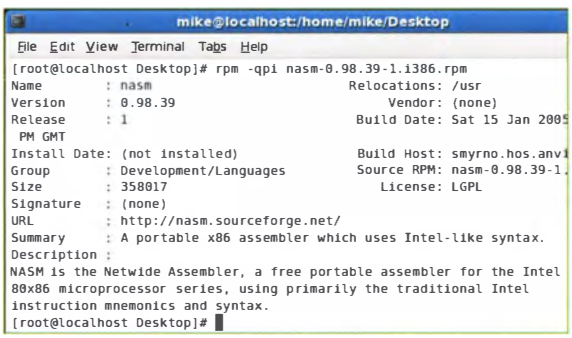
Sie sehen, dass die Struktur des Dateinamens die gleiche ist wie die bei den Debian-Paketen: Zuerst kommt der Name des Pakets, dann die Version (hier Version **0.98.39**), es folgt die eigene Versionsnummer des Paketbetreuers (**-1**), und am Ende steht der Hinweis auf die CPU-Architektur und der Zusatz **.rpm**.

Achten Sie auf die verwendeten Parameter in diesem Befehl: **-U** ist besonders wichtig, denn es bedeutet „Upgrade“. Mit **rpm -i** können Sie ein Paket installieren, aber es wird ihm nicht gefallen, wenn schon eine ältere Version auf dem System installiert ist. **-U** dagegen wird entweder ein neues Paket installieren oder ein schon existierendes aktualisieren, was bedeutet, dass Sie nur einen Befehl verwenden müssen.

Wenn Sie eine RPM-Datei heruntergeladen haben und nun überprüfen wollen, ob sie nicht beschädigt ist, geben Sie den Befehl **rpm –checksig filename** ein. Auch ein Paket wieder zu entfernen, geht problemlos – führen Sie einfach den Befehl **rpm -e nasm** durch.

Es gibt ein paar Möglichkeiten, wie Sie an Informationen über das Paket herankommen können. Wenn es um eine RPM-Datei geht, die Sie noch nicht installiert haben, dann geben Sie ein:

```
rpm -qpi nasm-0.98.39-1.i386.rpm
```



› Informationen über ein Paket erhalten Sie schnell und problemlos mit dem rpm q-Befehl.

Wenn die Pakete bereits auf dem Rechner installiert sind, entfernen Sie das **p**-Kennzeichen und benutzen Sie nur den Stamm des Paketnamens. Der obige Befehl würde, wenn *NASM* schon installiert ist, also folgendermaßen aussehen:

```
rpm -qR nasm
```

Wenn Sie eine Liste der Dateien benötigen, die von einem Paket installiert werden, verwenden Sie **rpm -ql nasm**. In welches Paket eine Datei gehört, erfahren Sie mit **rpm -qf /path/to/file**. Der **rpm**-Befehl ist außergewöhnlich vielseitig einsetzbar, ganz ähnlich wie sein Vetter **dpkg**. Wenn Sie noch mehr über seine vielen Möglichkeiten erfahren möchten, schauen Sie auf der Man-Page nach (**man rpm**).

Mit dem **rpm**-Befehl kann man sinnvoll mit lokalen Paketen arbeiten, doch es gibt ebenfalls ein Dienstprogramm, das das Finden und Aufrufen von Paketen und Abhängigkeiten im Internet automatisiert, ähnlich wie Debians APT. Bei RPM heißt das Paketverwaltungssystem *Yum – Yellowdog Updated Modifier –*, und es basierte ursprünglich auf einem Programm für eine andere Distribution.

Wenn wir z.B. die *Z Shell* installieren möchten, doch wir haben keine Pakete lokal auf unserem Rechnern, dann geben wir ein:

```
yum install zsh
```

Yum durchsucht daraufhin seinen Zwischenspeicher nach Paketinformationen, erfährt dabei, welche Abhängigkeiten benötigt werden und fordert Sie auf, Y zu drücken, wenn Sie mit der Operation fortfahren wollen. Nach der Bestätigung lädt *Yum* die erforderlichen Pakete herunter und installiert sie. Eine Liste der Pakete, die ein bestimmten Stichwort enthalten, können Sie mit **yum list**, gefolgt von dem Stichwort, aufrufen. Informationen über ein Paket vor der Installation erhalten Sie mit **yum info**, wieder gefolgt vom Name des Pakets.

Das Auffinden von Aktualisierungen von Betriebssystemen ist eine besondere Stärke von *Yum*. Geben Sie den Befehl **yum update** ein, und Sie erhalten eine Liste von allen Paketen, die seit dem Zeitpunkt ihrer Installation auch nur gering verändert wurden. Doch wo genau findet *Yum* diese Pakete? Die Antwort liegt in dem **/etc/yum/repos.d**-Verzeichnis. Darin befinden sich Textdateien mit der Endung **.repo**, die Adressen von Repositories enthalten. So enthält z.B. die Standard-Installation von CentOS Repositories für alle CentOS-Hauptpakete samt ihrer wichtigsten Updates.

In dieses Verzeichnis können Sie Ihre eigenen Dateien hinzufügen, wenn Sie im Internet ein Programm finden, das ein passendes Repository für Ihre Distributionsversion hat. Doch führen Sie danach auf jeden Fall den Befehl **yum makecache** aus, damit die lokal gespeicherten Daten aktualisiert werden. *Yum* ist extrem konfigurierbar – in der **/etc/yum.conf**-Datei finden Sie noch viele weitere Optionen, mit denen Sie herumexperimentieren können. ■

Quick-Tipp

Wenn Sie in einem auf Debian basierenden System eine Liste von allen installierten Paketen aufrufen möchten, geben Sie **dpkg -l** ein. In Distributionen, die auf RPM basieren, heißt der entsprechende Befehl **rpm -qa**. Da diese Listen sehr lang sein können, ist es wahrscheinlich weniger umständlich, wenn Sie die Ausgabe in eine Textdatei umleiten. Der Befehl dazu lautet: **rpm -qa > list.txt**.

Testen Sie Ihr Wissen!

Wenn Sie diese Lerneinheit einmal durchgearbeitet, alle Konzepte verstanden und Ihre eigenen Varianten der Befehle ausprobiert haben, lohnt sich ein kleiner Test, der Ihnen zeigt, ob Sie Fragen auch in einer Prüfungssituation beantworten können. Lösen Sie die folgenden Fragen.

Die Antworten stehen unten.

1. Mit welchem Befehl kann man ein Debian-Paket samt seiner Konfigurationsdateien entfernen?
2. Welche Datei enthält eine Liste von Repositories, die in auf Debian basierenden Distributionen verwendet werden?
3. Mit welchem Befehl kann man eine detaillierte Liste mit Informationen über ein Debian-Paket aufrufen?
4. Mit welchem Befehl würden Sie eine RPM-Datei in eine Deb-Datei konvertieren?
5. Wie entfernt man ein Paket aus einem auf RPM basierenden System?
6. Wo befinden sich die Repositories von *Yum*?
7. Wie aktualisiert man in *Yum* den Zwischenspeicher, in dem die Pakete abgelegt sind?

1 dpkg -purge 2 /etc/aps/sources.list 3 dpkg -s 4 alien -to-deb
5 rpm -e 6 /etc/yum.repos.de 7 yum makecache

Lernen Sie Linux

MIT MIKE SAUNDERS

Lektion 5: Nachdem wir jetzt ausführlich Hardware, Dateisystem und Pakete erforscht haben, wird es Zeit, dass wir die Befehlszeile meistern.

Linux Professional Institute

Leute, die es nicht besser wissen, halten die Befehlszeile für ein Relikt aus den 1970ern, einen sinnfreien Tummelplatz von Computerfreaks, wohin sich kein normaler Mensch verirrt. Betritt man aber die Welt eines Systemadministrators, wird einem klar, dass solche Unkenrufe völlig an den Tatsachen vorbeigehen. Die Befehlszeile, auch Shell genannt, ist wichtiger denn je, und das hat gute Gründe:

» Die Befehlszeile ist immer da. Sie liegt unter allen Ebenen der wundervollen GUI-Anwendungen, die wir bei einer typischen Desktop-Installation in Linux sehen. Deshalb können Sie, selbst wenn Ihr Window-Manager Ärger macht, STRG+ALT+F2 drücken, und es erscheint eine Eingabeaufforderung, mit der Sie das Problem beseitigen.

» Die Befehlszeile braucht keine grafische Benutzeroberfläche. Sie können sich (über SSH) von der anderen Seite des Planeten in den Remote-Rechner einloggen und damit genauso arbeiten, als säßen Sie an Ihrem eigenen Computer. Schwerfälliges VNC oder ein Remote Desktop sind überflüssig. Auch stören auf vielen Rechnern wie z.B. Servern all die Spielereien einer GUI. Mehr als die Befehlszeile brauchen Sie da nicht.

» Die Befehlszeile ist direkt. Sie führt genau das aus, was Sie eingegeben haben. Keine irren Anweisungen à la „Klicken Sie auf den Knopf links oben, finden Sie das Menü, das mit Foo bezeichnet ist, und machen Sie ein Kreuzchen ins Kästchen direkt daneben." In die Befehlszeile tippen Sie genau das ein, was Sie vom Rechner wollen, und der führt es dann aus. Kein Herumgemurkse, alles ist klar.

Deshalb verfügen alle guten Systemadministratoren über ein fundiertes Wissen, was die Funktion der Befehlszeile betrifft. In den vorherigen Lektionen haben wir schon einige Befehle losgelöst betrachtet, jetzt schauen wir uns *Bash* – die Standard-Shell in 99,9 % aller Linux-Distributionen – einmal im Detail an.

Teil 1: Erste Orientierung

Wenn Sie eine grafische Linux-Installation verwenden, können Sie die Befehlszeile über Ihre Desktop-Menüs aufrufen – meistens wird sie Terminal, Shell, *XTerm* oder *Konsole* genannt. In dieser Lerneinheit verwenden wir CentOS 5.5, wo Sie die Befehlszeile über Anwendungen > Zubehör > Terminal finden. Wenn Sie das Terminal geöffnet hat, sehen Sie Folgendes:

```
[mike@localhost ~]$
```

Die Eingabezeile besteht aus vier Teilen: Der erste ist der Name des gerade eingeloggten Users, hier **mike**. Dann kommt der Hostname des Rechners, den wir benutzen: **localhost**. Das Tilde-Zeichen (~) zeigt an, in welchem Verzeichnis wir gerade arbeiten. Wären wir z.B. in **/usr/bin**, dann würde an dieser Stelle **bin** stehen. Normalerweise startet eine Terminal-Sitzung im Basisverzeichnis des Nutzers, und die Tilde ist hier ein Kurzzeichen für **/home/mike**, weshalb sie in der Befehlszeile auftaucht. Und am Schluss steht das Dollarzeichen und fordert uns auf, etwas einzugeben. Das Dollarzeichen weist darauf hin, dass wir Befehle als ein normaler Nutzer ausführen. Wenn Sie **su** (und dann Ihr Passwort) eingeben, um zum Superuser-(root)-Account zu wechseln, verwandelt sich das Dollarzeichen in ein **#**-Zeichen.

Viele Befehle bestehen nur aus einem einzigen Wort. Geben Sie z.B. ein:

```
uname
```

Dadurch wird der Name des Betriebssystems ausgegeben, „Linux". Allerdings bietet **uname** noch mehr Funktionen; diese können durch den Einsatz von sogenannten Parametern aufgerufen werden. Parameter werden normalerweise über einen

Die Man-Pages

Sie möchten mehr erfahren über die Optionen, die ein Befehl oder ein Programm zu bieten hat? Die meisten Befehle haben eine Dokumentation in Form von Handbuchseiten oder Man-Pages (**man** von *manual*). Das sind keine benutzerfreundlichen Leitfäden, sondern es sind Seiten zum schnellen Nachschlagen, wenn Sie ganz gezielt nach einer bestimmten Option suchen. Auf diese Seiten greifen Sie mit dem **man**-Befehl zu, gefolgt von dem Befehl, um den es Ihnen gerade geht, z.B. **man ls**. Im Viewer können Sie mit den Pfeiltasten nach oben und unten scrollen. Wenn Sie im Hilfsseitentext nach einem speziellen Wort suchen möchten, drücken Sie die Slash-Taste (/) und geben dann das gesuchte Wort ein.

Bindestrich mit einem Buchstaben oder Wort spezifiziert. Geben Sie einmal diesen Befehl ein:

```
uname -a
```

Er führt das **uname**-Programm aus, gibt jedoch den Parameter **-a** vor, das „zeige alle Informationen" bedeutet – Sie bekommen so einen ausführlicheren Output. Mit dem **--help**-Parameter können Sie bei den meisten Befehlen einsehen, welche Optionen zur Verfügung stehen, hier z.B. **uname --help**.

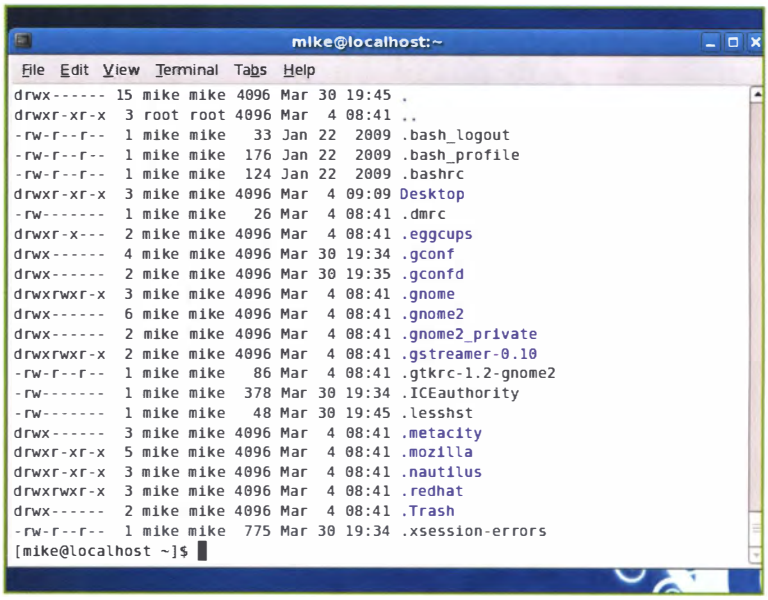
Nun wissen Sie also, was genau Sie mit der Befehlszeile vor sich haben, wie Sie einen Befehl eingeben und wie Sie seine Parameter ändern.

Weiter geht es mit Datei-Management. Geben Sie zunächst **ls** ein – für *list files*, „Dateien auflisten". Der Befehl zeigt die Dateien und Verzeichnisse im aktuellen Verzeichnis an, und je nachdem, welches System Sie verwenden, markiert er die verschiedenen Elemente wie Unterverzeichnisse sogar mit Farben.

Der **ls**-Befehl allein zeigt Ihnen keine versteckten Elemente an – also Dateien und Verzeichnisse, die mit einem Punkt beginnen. Wenn Sie aber **ls -a** eingeben, sehen Sie alles (versteckte Dateien werden meistens für Konfigurationsdateien verwendet, die Ihnen aber nicht die Ausgabe zumüllen sollen). Wenn Sie eine detaillierte Liste möchten, geben Sie **ls -l** ein. Sie können mehrere Kennzeichen kombinieren, wie z.B. **ls -l -a**, oder noch schneller **ls -la**. Dadurch erhalten Sie noch ausführlichere Informationen, etwa den Besitzer einer Datei, ihre Größe, das Modifikationsdatum und vieles mehr.

Bis jetzt sind wir nur in unserem Basisverzeichnis tätig, aber im normalen Leben muss ein Administrator sich auch in anderen Verzeichnissen herumtreiben. Doch zunächst legen wir ein neues Verzeichnis an.

```
mkdir newdir
```



› Mit dem ls-Befehl rufen Sie Dateilisten auf. Er ist vielseitig einsetzbar. So kann er Dateien auf alle möglichen Arten darstellen, wie z.B. in einer sehr ausführlichen Liste, wie Sie sie hier sehen.

In dieses Verzeichnis wechseln wir mit dem **cd**-Befehl (*change directory*, also „wechsle das Verzeichnis"):

```
cd newdir
```

Wenn Sie jetzt **ls** eingeben, sehen Sie, dass das Verzeichnis leer ist. Um wieder in das vorherige Verzeichnis zu gelangen, geben Sie **cd ..** ein. Falls Sie früher DOS verwendet haben, wird Ihnen das bekannt vorkommen – **..** bezieht sich immer auf das Verzeichnis über demjenigen, in dem man sich gerade befindet. Doch anders als in DOS ist die Leerstelle Teil des Befehls. Und beachten Sie, dass – ebenfalls anders als in DOS – hier bei allen Befehlen und Dateinamen zwischen Groß- und Kleinschreibung unterschieden wird. So können Sie den Befehl **cd** mit Verzeichnissen verwenden, aber Sie können auch ganze Pfade angeben. Beispielsweise wechseln Sie in das **/usr/bin**-Verzeichnis mit dem Befehl:

```
cd /usr/bin
```

Es gibt noch eine weitere praktische Funktion des **cd**-Befehls: Wenn Sie nur **cd** eingeben, gelangen Sie zurück in Ihr Basisverzeichnis. Das spart Ihnen Zeit, vor allem, wenn Sie einen langen Login-Namen haben.

Wollen Sie den vollständigen Pfad des Verzeichnisses sehen, in dem Sie sich gerade befinden, dann geben Sie den Befehl **pwd** (für *print working directory*) ein. Wenn Sie gerade das Verzeichnis gewechselt haben, also z.B. von **/usr/bin** zu **/etc** gegangen sind, dann geben Sie einfach **cd -** ein, um wieder in das Verzeichnis zu gelangen, in dem Sie eben waren. »

Der $PATH in die Freiheit

Es gibt eine besondere Variable namens **$PATH**, die eine Liste von Speicherorten enthält, von denen aus Sie Programme laufen lassen können. Rufen Sie mit **echo $PATH** diese Verzeichnisse auf, dort sehen Sie **/usr/bin**, **/usr/sbin** u.ä. Wenn Sie einen Befehl eingeben, beispielsweise **nano**, um den *Nano*-Texteditor zu öffnen, dann sucht die Shell in diesen Speicherplätzen nach dem Programm. Allerdings ist das aktuelle Verzeichnis nie Teil von **$PATH**. Das ist eine Sicherheitsmaßnahme, damit Trojaner (etwa eine bösartige **ls**-Binärdatei) nicht in Ihr Basisverzeichnis gelangen und dort jedes Mal ausgeführt werden, wenn Sie **ls** ausführen. Wollen Sie ein Programm aus Ihrem aktuellen Verzeichnis starten, dann stellen Sie ihm Punkt-Slash voran, also z.B. **./myprog**. Das mag sich lästig anhören, aber es hat sich im Lauf der Jahre als ein großer Pluspunkt in der Systemsicherheit von Linux und Unix erwiesen. Es kann sein, dass Sie etwas in **/opt** installiert haben, das Sie zu Ihrer **$PATH**-Variable hinzufügen müssen, damit sie fehlerfrei funktioniert. Benutzen Sie dafür den **export**-Befehl, wie er oben im Text erklärt wird. Doch wir wollen natürlich nicht alle schon in **$PATH** aufgelisteten Speicherplätze überschreiben. Geben Sie deshalb den Befehl so ein:

```
export PATH=$PATH:/opt/newprog
```

Wenn Sie jetzt **echo $PATH** ausführen, erscheinen die vorherigen Speicherplätze, und **/opt/newprog** wird ans Ende der Liste angefügt.

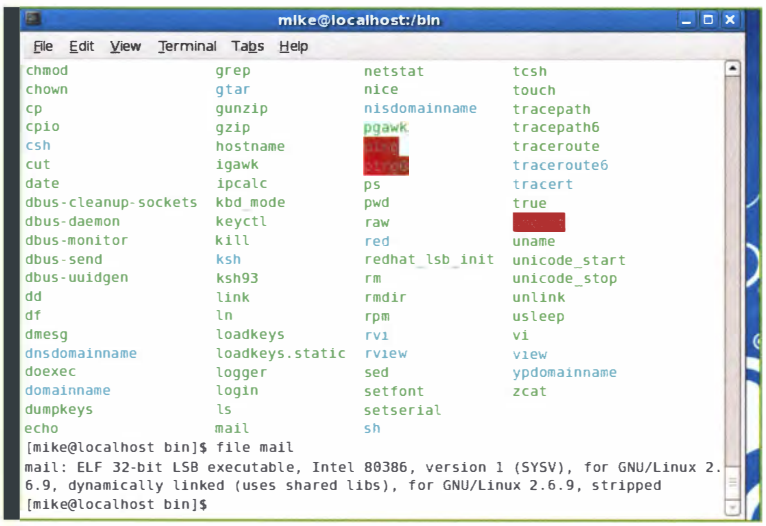
Teil 2: Wir tauchen noch tiefer in die Materie ein

Quick-Tipp

Sie wollen einen Befehl eingeben und die Shell-Sitzung direkt beenden (bzw. das Terminal schließen), nachdem er ausgeführt wurde? Verwenden Sie dafür den **exec**-Befehl, z.B. **exec nano**.
Sobald Sie den Texteditor *Nano* verlassen haben, schließt sich auch das Fenster des Terminals.

Verzeichnis- und Dateinamen können ziemlich lang werden, besonders dann, wenn die einzelnen Elemente als Pfade aneinandergereiht werden. Doch *Bash* bietet eine clevere Funktion: die Namensvervollständigung (*tab completion*). Tippen Sie einfach die ersten Buchstaben einer Datei oder eines Verzeichnisses, drücken Sie auf Tab, und *Bash* vervollständigt den Namen. Versuchen Sie es und geben Sie z.B. **cd /usr/lo** ein und drücken Sie dann auf Tab – die Zeile wird zu **/usr/local** ergänzt. Falls zwei oder mehr Verzeichnisse in **/usr** liegen, die mit **lo** beginnen, dann zeigt Ihnen *Bash* an, welche Verzeichnisse in Frage kommen.

Mit der Namensvervollständigung können Sie in Ihrem Linux-Leben viel Zeit sparen, das gleiche gilt für den Befehlsverlauf. Mit den Pfeiltasten können Sie nach oben und unten durch die vorangegangenen Befehle navigieren (diese werden in **.bash_history** in Ihrem Basisverzeichnis gespeichert). Mit den Pfeiltasten können Sie sich auch nach rechts und links durch die Befehle bewegen und sie ändern. Geben Sie **history** ein, damit rufen Sie eine Liste der zuletzt eingegebenen Befehle auf.



Was tun, wenn Sie nicht wissen, welches Format eine Datei hat? Mit dem file-Befehl können Sie dies in Sekundenschnelle feststellen. Mit dem Befehl werden die ersten Bytes der Datei überprüft und daran das Format erkannt.

Schauen wir uns jetzt einige Befehle zur Dateimanipulation an. Eine Datei kopieren können Sie mit dem Befehl **cp**, also z.B.

```
cp file1.txt file2.txt
```

Mit dem Befehl **cp file1 file2 dir** kopieren Sie gleich mehrere Dateien in ein Verzeichnis. Ähnlich funktioniert der Befehl **mv**, um Dateien an einen neuen Ort zu verschieben. Mit diesem Befehl können Sie Dateien auch umbenennen, und zwar so: **mv oldname newname**. Um eine Datei zu entfernen, geben Sie **rm filename** ein. Aber Vorsicht, **rm** dringt nicht weit in die Verzeichnisse vor, schon gar nicht in die Unterverzeichnisse. Für eine vollständige Löschung benutzen Sie den Parameter **-r** für rekursives Löschen:

```
rm -r directory
```

Dadurch wird das gesamte Verzeichnis entfernt, sämtliche darin liegende Dateien und alle Unterverzeichnisse – es ist ein sehr wirksamer und zerstörerischer Befehl! Wenn Sie auf eine Datei stoßen, die Sie nicht zuordnen können – etwa, weil der Dateiname nicht aussagekräftig ist oder die Datei keine sinnvolle Erweiterung besitzt – dann geben Sie den **file**-Befehl ein:

```
file /usr/bin/emacs
```

Emacs ist ein ausgezeichnetes Dienstprogramm, das die ersten Bytes einer Datei prüft, um festzustellen, um welches Format es sich handelt. Wenn Emacs einen JPEG-Header entdeckt, dann wird es Ihnen sagen, dass es sich bei der Datei um eine JPEG-Datei handelt. Der **file**-Befehl greift hier auf ein System namens „magic“ zu. Es ist eine Datenbank mit Byte-Sequenzen, nach denen man in Dateien suchen muss, wenn man ihr Format bestimmen will. Diese Bestimmung ist nicht immer 100 % korrekt, und es kann schon mal vorkommen, dass eine einfache Textdatei als „Microsoft FoxPro Datenbank“ oder etwas ähnlich Absurdes identifiziert wird, nur weil die Datei zufällig eine bestimmte Sequenz von Bytes enthält.

Aus irgendwelchen Gründen wollen Sie vielleicht den Zeitstempel einer Datei aktualisieren oder eine leere Datei anlegen. Dafür ist der **touch**-Befehl wie geschaffen. Bestimmt werden Sie oft von der Befehlszeile aus Dateien suchen wollen. Das können Sie auf zweierlei Weise tun: mit **locate** oder **find**. Die Befehle klingen ähnlich, aber es gibt einen entscheidenden Unterschied: Wenn Sie **locate foobar.txt** eingeben, wird eine Anfrage bei einer zuvor erstellten Datenbank aller Dateien im System durchgeführt, und Sie erfahren blitzschnell, wo die gesuchte

Archive erstellen und erweitern

Software, Patches und andere Pakete sind meistens komprimierte Dateien. Es gibt etliche Komprimierungsformate, doch zum Glück beinhalten die meisten Linux-Distributionen die nötigen Tools, um die Pakete zu entpacken und erneut zu komprimieren. Allerdings verwenden nicht alle die gleichen Parameter. Hier ein kleiner Überblick:

- **.gz** Eine einzelne komprimierte Datei. Entpacken Sie sie mit **gunzip foo.gz**. Um eine Datei zu komprimieren, benutzen Sie **gzip foo**.
- **.bz2** Wie **.gz**, aber mit einer stärkeren (und langsameren) Komprimierung. Wird mit **bunzip2** entpackt. Um eine Datei wieder zu komprimieren, verwenden Sie **bzip2**. Auf älteren Rechnern lief dieses Format nur mit Schwierigkeiten, doch auf den heutigen PCs wird es bevorzugt eingesetzt, um große Quellcode-Archive wie den Linux-Kernel zu verteilen.
- **.tar** Ein Bandarchivsystem. Heute verwenden nur noch wenige Leute Datenbänder, aber es ist ein System, mit dem man viele Dateien (ohne Komprimierung) in einer einzelnen Datei bündeln kann. Entpackt wird ein **.tar**-Archiv mit **tar xfv foo.tar**. Dateien bündeln kann man mit dem Befehl **tar cfv foo.tar file1 file2** – er erstellt ein neues Archiv mit der Bezeichnung **foo.tar**, das die aufgelisteten Dateien enthält.
- **.tar.gz / .tar.bz2** Eine Kombination der vorigen Formate und die gebräuchlichste Methode, Quellcode zu verteilen. Die Dateien werden mit **tar** zu einem einzigen Knäuel zusammengeführt und dann mit **gzip** oder **bzip2** komprimiert. Entpackt wird mit den folgenden Befehlen: **tar xfv filename.tar.gz** oder **tar xfv filename.tar.bz2**. Das Komprimieren erfolgt über die Befehle: **tar cfvz foo.tar.gz file1 file2** (für **.tar.gz**) oder **tar cfvj foo.tar.bz2 file1 file2** (für **.tar.bz2**).
- **.cpio** Ein selteneres Format, das ebenfalls mehrere Dateien zu einer einzelnen Datei zusammenführt (ohne Kompression). Entpacken Sie die Dateien mit **cpio -id <filename>**. Die Dateibündelung führen Sie mit dem Befehl **ls file1 file2 | cpio -ov > foo.cpio** aus. Sie werden CPIO-Dateien mit Sicherheit begegnen, wenn Sie mit initrd-Images arbeiten.
- Ebenfalls ein nützliches Dienstprogramm ist **dd**, das Daten von einer Quelle in eine andere kopiert. Das Programm ist besonders hilfreich, wenn Sie Disk-Images von einem physischen Datenträger extrahieren. Wenn Sie also z.B. eine CD oder DVD in das entsprechende Laufwerk schieben und dann **dd if=/dev/cdrom of=myfile.iso** eingeben, erhalten Sie ein ISO-Abbild (das Sie dann verteilen oder brennen können).

Datei sich befindet. Diese Datenbank wird allerdings normalerweise nur einmal am Tag aktualisiert, von dem Dienstprogramm *Cron*, weshalb sie nicht immer dem aktuellen Stand entspricht.

Für Suchen nach aktuell benutzten Dateien verwenden Sie besser den **find**-Befehl, z.B. so:

```
find /home/mike -name hamster
```

Dadurch wird im Verzeichnis **/home/mike** (und allen Unterverzeichnissen) eine gründliche Suche nach all den Elementen gestartet, die „hamster" im Namen haben. Was aber, wenn Sie das aktuelle Verzeichnis durchsuchen möchten, ohne den vollständigen Pfad einzugeben? Oben haben wir davon gesprochen, dass .. immer das Verzeichnis oberhalb des aktuellen meint. Nun, . – also ein Punkt – bezieht sich auf das aktuelle Verzeichnis. Sie können also den **find**-Befehl verkürzen, vorausgesetzt, Sie befinden sich im Verzeichnis **/home/mike**, also:

```
find . -name hamster
```

Der **find**-Befehl kann auch für Dateigrößen angewendet werden: **find . -size +100k** lokalisiert alle Dateien im aktuellen Verzeichnis, die größer als 100 Kilobytes sind (für Megabytes benutzen Sie M, für Gigabytes G). Eine weitere Möglichkeit ist die Suche nach Dateityp: **find . -type f** zeigt Ihnen ausschließlich Dateien, während **find . -type d** nur die Verzeichnisse aufruft. Für sehr spezifische Dateisuchen können Sie die Parameter **-name, -size und -type** auch verbinden. Das *-Zeichen bedeutet „jede Kombination von Buchstaben, Ziffern und anderen Zeichen". Schauen Sie sich diesen Befehl an:

```
ls *.jpg
```

Er ruft alle Dateien mit einer **.jpg**-Erweiterung auf, egal ob die Datei **bunnyrabbit.jpg**, **4357634.jpg** oder sonst wie heißt. Dieser Befehl ist ausgesprochen nützlich, wenn man Dateien verschieben oder löschen möchte. Stellen Sie sich vor, Sie haben ein Verzeichnis voller Bilder und würden gerne diejenigen mit der Erweiterung **.bmp** löschen. Mit dem Befehl **rm *.bmp** ist das ganz einfach. Wenn Sie das Platzhalterzeichen nur für einen einzigen Buchstaben einsetzen wollen, dann verwenden Sie ein Fragezeichen, so wie hier:

```
mv picture?.jpg mypics
```

Dieser Befehl verschiebt **picture1.jpg**, **pictureA.jpg** usw. in das **mypics**-Verzeichnis.

Quick-Tipp

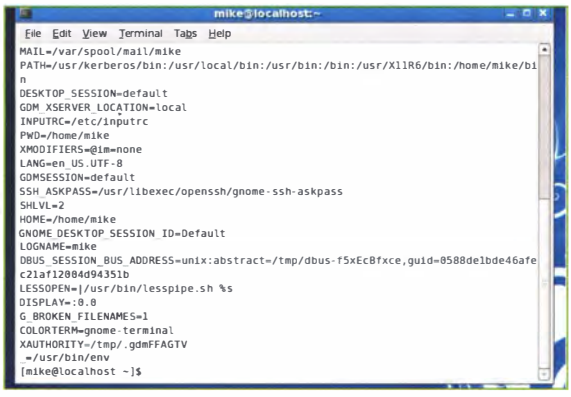
Falls es bei einem Befehl so aussieht, als würde die Durchführung noch Stunden dauern, und Sie möchten ihn abbrechen, dann drücken Sie STRG+C. Beachten Sie, dass der Befehl sofort unterbrochen wird, deshalb werden die Dateien, auf die er gerade zugegriffen hat, nicht ordnungsgemäß gelöscht, geschlossen oder zurückgesetzt.

Teil 3: Die Umgebung verstehen

Normalerweise gibt man an der Befehlszeile einen Befehl nach dem anderen ein. Doch diese Befehle sind abhängig von der Umgebung, in der sie ausgeführt werden. Sogenannte Umgebungsvariablen speichern Informationen über Optionen und Einstellungen, und Programme können dann auf diese Informationen zugreifen, um ihre Arbeitsweise festzulegen. Umgebungsvariablen werden normalerweise in Großbuchstaben geschrieben und beginnen mit einem Dollarzeichen. Versuchen Sie es einmal damit:

```
echo $BASH_VERSION
```

Der Befehl **echo** dient nur dazu, Text auf dem Bildschirm auszugeben, in diesem Fall den Inhalt der **$BASH_VERSION**-Umgebungsvariable. Eine Nummer erscheint, so etwas wie 3.2.25.



› Umgebungsvariablen verändern die Art und Weise, in der Programme ablaufen – mit dem env-Befehl erhalten Sie eine vollständige Liste der Variablen.

Programme können auf diese Variable zugreifen, um zu erfahren, ob ein Nutzer Version 3.0 oder höher in Betrieb hat und ob somit bestimmte Funktionen zur Verfügung stehen. Eine vollständige Liste aller gerade benutzten Umweltvariablen erhalten Sie mit dem Befehl **env**. Ihre eigenen Umweltvariable erstellen Sie so:

```
export FOO="bar"
echo $FOO
```

Diese neue **$FOO**-Variable existiert nur so lange, wie die Terminal-Sitzung geöffnet ist. Wenn Sie die Sitzung beenden – also **exit** eingeben oder STRG+D drücken – geht die neue Variable verloren. Um das zu verhindern, müssen Sie in Ihrem Basisverzeichnis die Textdatei **.bashrc** verändern, die Variablendefinitionen und andere Einstellungen enthält, die gelesen werden, wenn eine Sitzung mit der Befehlszeile startet. Speichern Sie Ihre Änderungen, starten Sie das Terminal erneut, und die frisch erstellte Variable ist da.

Bash hat noch andere Variablen außen denen für die Umgebung, nämlich seine eigenen. Geben Sie **set** ein, und Sie erhalten eine vollständige Liste. Wenn Sie eine Variable entfernen wollen, entweder eine für *Bash* oder die Umgebung, dann können Sie dies mit folgendem Befehl tun:

```
unset FOO
```

Diese Funktionen, zusammen mit der Namensvervollständigung, dem Platzhalterzeichen und dem **history**-Befehl, erlauben eine außerordentlich effiziente Arbeit mit der Linux-Befehlszeile. Es ist kein Vergleich mehr zu der schwerfälligen DOS-Eingabeaufforderung von früher. Glauben Sie mir, wenn Ihnen die Befehlszeile erst einmal vertrauter ist, werden Sie bald den Dateimanager über Bord werfen wollen. ■

Quick-Tipp

Ihnen ist aus Versehen die Ausgabe auf Ihrem Terminal durcheinandergeraten? Geben Sie **clear** ein (oder drücken Sie STRG+L), um den Bildschirm aufzuräumen. Wenn das nicht funktioniert und seltsame Zeichen erscheinen, weil das Terminal nur noch zufällige Binärdaten ausspuckt, dann versuchen Sie es mit **reset**.

Testen Sie Ihr Wissen!

Was meinen Sie, haben Sie die Themen und Befehle dieser Lektion verinnerlicht? Versuchen Sie sich mal an den Antworten zu folgenden Fragen.

1. Was bedeutet das Tilde-Zeichen (~) in einer Eingabeaufforderung?
2. Wie erhalten Sie eine Liste aller Dateien im aktuellen Verzeichnis, mit ausführlichen Informationen?
3. Welcher Befehl, mit dem man Dateien suchen kann, greift auf eine zuvor angelegte Datenbank zu?
4. Wie würden Sie die Umgebungsvariable **$WM** in **icewm** einstellen?
5. Wie fügen Sie **/opt/kde/bin** zu Ihrer **$PATH**-Variable hinzu?
6. Wie erstellen Sie ein **.tar.bz2**-Archiv aus dem Verzeichnis **myfiles**?
7. Sie möchten eine Version von *Nano* aus Ihrem aktuellen Verzeichnis, aber nicht mit Ihrer **$PATH**-Variable starten. Wie machen Sie das?

1 Basisverzeichnis 2 ls -la 3 locate 4 export WM="icewm" 5 export PATH=$PATH:/opt/kde/bin 6 tar cfvj archive.tar.bz2 myfiles 7 ./nano

Lernen Sie Linux

MIT MIKE SAUNDERS

Lektion 6: Die Grundlagen der Befehlszeile haben Sie jetzt drauf, dann können wir uns fortgeschrittenen Tricks und Techniken zuwenden.

Linux Professional Institute

In der letzten Lektion haben wir erfahren, dass es keineswegs umständlich ist, mit einem Computer über das Terminal zu kommunizieren, und dass die Befehlszeile durch GUIs auch nicht völlig überholt ist. Im Gegenteil, sie ist ein extrem flexibles und leistungsstarkes Interface, mit dem man in Sekunden Aufgaben bewältigt, für die man sonst Hunderte von Mausklicks braucht. Außerdem können Sie sich nicht darauf verlassen, dass Ihr X Window-System stets richtig funktioniert – und für solche Fälle ist es notwendig, dass man sich mit der Befehlszeile auskennt. Wenn Linux als Betriebssystem eines Servers läuft, dann wollen Sie sowieso kein schwergewichtiges GUI auf Ihrer Festplatte haben.

Nach den Grundfunktionen der Befehlszeile einschließlich der Editierbefehle, der Verwendung des Platzhalterzeichens und der Möglichkeiten der Dateimanipulation kommen wir nun zu einigen fortgeschrittenen Themen.

Teil 1: Das Umleiten der Ausgabe

Wenn Sie mit der Befehlszeile arbeiten, möchten Sie in den allermeisten Fällen, dass das Ergebnis Ihrer Befehle auf dem Bildschirm ausgegeben wird. Doch der Bildschirm hat nichts Magisches an sich, und was Unix betrifft, gleicht er jedem anderen Gerät. Mehr noch, getreu der Unix-Philosophie „Alles ist eine Datei“ kann die Ausgabe von Befehlen auch zu Dateien anstatt zum Bildschirm geschickt werden. Geben Sie diesen Befehl ein:

```
uname -a > output.txt
```

Wie wir in Lektion 5 gelernt haben, werden mit **uname -a** Informationen über das aktuell installierte Betriebssystem aufgerufen. Standardmäßig gibt der Befehl diese Informationen auf dem Bildschirm aus, doch durch das Größer-als-Zeichen **>** wird die Ausgabe nicht auf dem Bildschirm angezeigt, sondern in die Datei **output.txt** umgeleitet. Sie können die Datei jetzt in Ihrem Texteditor öffnen oder sie mit **cat output.txt** auf dem Bildschirm anzeigen. Probieren Sie einmal diesen Befehl:

```
df > output.txt
```

Schauen Sie sich den Inhalt der Datei **output.txt** an: Sie haben das Ergebnis des Speicherverbrauch-Befehls **df** vor sich. Beachten Sie, dass die Dateiinhalte von **output.txt** hiermit überschrieben wurden, Sie finden keine Spur mehr von dem Befehl **uname -a** von vorhin. Wenn Sie die Ausgabe eines Befehls in der Datei hinzufügen möchten, ohne deren Inhalt zu löschen, dann geben Sie das so ein:

```
uname -a > output.txt
df >> output.txt
```

Das doppelte Größer-als-Symbol **>>** in der zweiten Zeile bedeutet „hinzufügen statt überschreiben“. Auf diese Weise können Sie die Ergebnisse einer ganzen Reihe von Befehlen in einer einzigen Ausgabedatei anzeigen.

So funktioniert also das Umleiten. Man kann jedoch die Ausgabe eines Befehls auch direkt zu einem anderen Programm schicken. Eine solche Datenübergabe wird als Pipe bezeichnet. Angenommen, Sie möchten sich die Ausgabe eines langen Befehls wie **ls -la** ansehen. Mit der gerade besprochenen Umleitmethode könnten Sie das so machen:

```
ls -la > output.txt
less output.txt
```

Dadurch wird die Liste in die Datei gesandt, wo Sie sie mit dem **less**-Tool ansehen können. Scrollen Sie mit den Pfeiltasten durch die Datei, und verlassen Sie sie mit Q. Das Ganze geht

Was sind reguläre Ausdrücke?

Auf den ersten Blick ist nichts „regulär“ an einem regulären Ausdruck. Im Gegenteil, wenn Ihnen so etwas wie

```
a\(\(b\)*\2\)*d
```

begegnet, dann würden Sie vermutlich gern das Weite suchen. Reguläre Ausdrücke dienen dazu, Textstücke zu identifizieren, und sind sehr kompliziert. Was immer Sie auch vorhaben – alle Wörter suchen, die mit drei Großbuchstaben beginnen und einer Zahl enden, oder alle Textstellen herausfiltern, die von Trennstrichen umgeben sind –, es gibt einen regulären Ausdruck, der genau das tut. Reguläre Ausdrücke sehen meistens aus wie reines Kauderwelsch, und dicke Bücher sind über sie verfasst worden, deshalb keine Sorge, wenn sie Ihnen Bauchschmerzen bereiten. Zum Glück brauchen Sie für die LPI-Schulung nur grob zu wissen, worum es geht. Wenn Sie mit regulären Ausdrücken in Berührung kommen, dann wahrscheinlich nur, wenn es um einen solchen geht, der Text ersetzt – normalerweise in Verbindung mit **sed**, einem Texteditor für Datenströme. Der Editor liest eine Eingabedatei, bearbeitet sie direkt und versendet dann den Output. Sie können **sed** mit einem regulären Ausdruck verwenden, um Text zu ersetzen, und zwar so:

```
cat file.txt | sed s/apple/banana/g > file2.txt
```

Dadurch wird der Inhalt von **file.txt** zu **sed** gesandt, mit der Aufforderung, einen ersetzenden regulären Ausdruck zu verwenden, der überall das Wort **apple** in **banana** ändert. Dann wird die Ausgabe in eine neue Datei umgeleitet. Das ist bei Weitem die häufigste Verwendung, wie Administratoren reguläre Ausdrücke einsetzen.

jedoch auch einfacher und ohne, dass man eine eigene Datei dafür braucht:

```
ls -la | less
```

Das darin vorkommende Pipe-Zeichen **|** ist in gedruckter Form nicht immer gut zu erkennen, und seine Position auf dem Keyboard kann je nach Tastaturlayout variieren. Das Pipe-Zeichen vermittelt der Shell, dass wir die Ausgabe eines Befehls zu einem anderen Befehl schicken möchten – in diesem Fall die Ausgabe von **ls -la** direkt zu **less**. Anstatt eine Datei zu lesen, liest **less** jetzt die Ausgabe des Programms vor der Pipe.

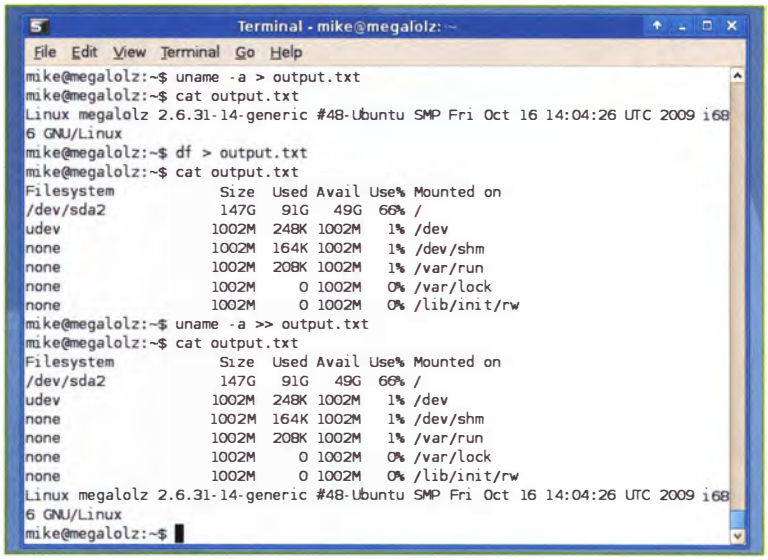
In bestimmten Situationen möchten Sie vielleicht die Ausgabe eines Befehls als eine Reihe von Argumenten für einen anderen Befehl nutzen. Vielleicht möchten Sie, dass *Gimp* alle JPEG-Bilder im aktuellen Verzeichnis und den Unterverzeichnissen öffnet. In der ersten Phase einer solchen Operation wird eine Liste aller Bilder zusammengestellt, und zwar mit dem **find**-Befehl:

```
find . -name "*.jpg"
```

Diese Information können wir nicht direkt per Pipe zu *Gimp* umleiten, weil durch die Pipe nur Rohdaten geschickt werden, während *Gimp* erwartet, dass die Dateinamen als Argumente ausgewiesen sind. Das machen wir mit **xargs**, einem ausgesprochen nützlichen Dienstprogramm, das aus dem Quellmaterial Argumentlisten zusammenstellt und sie an das Programm weiterleitet. Der Befehl dafür lautet:

```
find . -name "*.jpg" | xargs gimp
```

Sollten Sie die Ausgabe eines Befehls zwar auf dem Bildschirm angezeigt bekommen, aber zusätzlich auch in eine Datei umleiten wollen, erreichen Sie dies mit der **tee**-Utility:



› Wenn Sie Output in eine neue Datei umleiten (oder an bestehende Dateien anhängen) möchten, dann tun Sie das mit den Operatoren > und >>.

```
free -m | tee output.txt
```

Nun ist die Ausgabe des Befehls **free -m** (welcher Speicherplatz in Megabytes anzeigt) auf dem Bildschirm zu sehen, und sie wird außerdem an die Datei **output.txt** geschickt, wo Sie sie später lesen können. Hängen Sie noch die Option **-a** an den **tee**-Befehl, dann wird die Ausgabe zur Datei hinzugefügt und diese nicht überschrieben.

Teil 2: Text verarbeiten

Unix war schon immer ein fantastisches Betriebssystem, um mit Text umzugehen (sowohl in Dateien als auch bei der Weiterleitung mittels Pipes), und Linux setzt dies fort. Die meisten Distributionen enthalten eine große Anzahl von GNU-Dienstprogrammen, mit denen man Textströme manipulieren kann, indem man eine Menge von Zeichen benutzt, die man in viele verschiedene Formate umorganisieren und umstellen kann. Oft werden die Zeichen zusammen mit dem praktischen Pipe-Symbol verwendet. Wir kommen jetzt zu den wichtigsten Tools zur Textmanipulation, über die Sie für die LPI-Zertifizierung Bescheid wissen sollten.

Zunächst schauen wir uns an, wie man einen Textstrom erzeugt. Angenommen, Sie haben eine Datei namens **words.txt**, die den Text **Foo bar baz** enthält. Mit dem Befehl

```
cat words.txt
```

wird Ihnen der Inhalt der Datei auf dem Bildschirm angezeigt. **cat** steht für *concatenate* (verknüpfen) und kann, wie oben besprochen, mit Umleitungen oder dem Pipe-Symbol verwendet werden. Oft möchten Sie nur einen Teil der Ausgabe sehen, die durch diesen Befehl erzeugt wird. Mit dem **cut**-Befehl können Sie die Ausgabe auf einem bestimmten Abschnitt beschränken:

```
cat words.txt | cut -c 5-7
```

Die Ausgabe des **cat**-Befehls wird so zum **cut**-Befehl geleitet, mit der Aufforderung, die Zeichen 5 bis einschließlich 7 auszuschneiden. Beachten Sie, dass Leerstellen auch Zeichen sind, weshalb wir in diesem Fall als Ergebnis **bar** erhalten. Dieses Vorgehen ist allerdings eher ungewöhnlich, und Sie wollen wahrscheinlich selten genau das Wort ausschneiden, das mit dem fünften Zeichen des Texts beginnt (und drei Zeichen lang ist). Zum Glück kann **cut** den Text auf verschiedene Arten aufteilen. Schauen Sie sich diesen Befehl an:

```
cat words.txt | cut -d " " -f 2
```

»

Quick-Tipp

Sie möchten Ihre gesamte Arbeit an der Befehlszeile in einer Datei speichern? Geben Sie **script** ein, damit starten Sie eine neue Shell-Sitzung innerhalb der bereits laufenden. Führen Sie Ihre Befehle durch, tippen Sie dann **exit**. Eine Datei namens **typescript** wurde erstellt, die sämtliche Ausgaben Ihrer Befehle enthält.

Quick-Tipp

Wenn Sie bei irgendeinem Befehl Hilfe brauchen, schauen Sie auf der entsprechenden Man-Page nach. Mit **man cut** rufen Sie z.B. die Seite für den **cut**-Befehl auf. Sie können mit den Pfeiltasten durch die Seiten scrollen. Wenn Sie ein bestimmtes Wort suchen, drücken Sie den Schrägstrich (/) und geben Sie das Wort ein. Die Handbuchseite verlassen Sie mit Q.

» Hier fordern wir **cut** auf, dass es Leerstellen als Begrenzungszeichen nutzt – also die Zeichen, mit denen Felder im Text voneinander getrennt werden – und uns dann das zweite Feld im Text anzeigt. Weil unser Text **Foo bar baz** heißt, ist das Ergebnis hier **bar**. Ändern Sie die letzte Ziffer des Befehls zu 1, dann bekommen Sie **Foo**, oder zu 3, dann wird **baz** angezeigt. Damit können Sie genaue Positionen in einer einzelnen Textzeile ausschneiden.

Doch was ist, wenn Sie die Anzahl der ausgegebenen Zeilen beschränken möchten? Das machen wir mit den Utilities **head** und **tail**. Angenommen, Sie möchten die fünf größten Dateien im aktuellen Verzeichnis auflisten. Dazu können Sie sich mit **ls -lSh** eine nach Größe sortierte Liste ausgeben lassen. Allerdings wird diese Liste alle Dateien enthalten, und bei einem großen Verzeichnis kann das recht unübersichtlich werden. Mit dem **head**-Befehl können wir die Ausgabe einengen:

```
ls -lSh | head -n 6
```

Dadurch wird **head** aufgefordert, die Ausgabe auf die obersten sechs Zeilen zu beschränken. Von diesen gibt eine den Gesamtumfang aller Dateien an, uns interessieren die fünf Dateinamen darunter. Der direkte Gegenspieler dieses Befehls ist **tail**, der das Gleiche bewerkstelligt, aber am unteren Ende des Textstroms. Der Befehl

```
cat /var/log/messages | tail -n 5
```

zeigt die fünf letzten Zeilen von **/var/log/messages** an. Die Utility **tail** hat noch eine besonders nützliche Funktion. Man kann damit beobachten, wann eine Datei aktualisiert wird, und dies dann anzeigen lassen. Der Befehl dafür heißt **follow** und wird so verwendet:

```
tail -f /var/log/messages
```

Der Befehl zeigt Ihnen jedes Update der Log-Datei und setzt erst wieder aus, wenn Sie STRG+C drücken.

Wenn Sie mit großen Textmengen arbeiten, ist es sinnvoll, diese vor jeder Art von Bearbeitung erst einmal zu sortieren. Praktischerweise gibt es dafür in jeder normalen Linux-Installation einen **sort**-Befehl. Um diesen Befehl in Aktion zu erleben, erstellen Sie zunächst eine Datei mit dem Namen **list.txt** und den folgenden Inhalten:

```
ant
bear
dolphin
ant
bear
```

Führen Sie **cat list.txt** durch, und Sie erhalten erwartungsgemäß diese Liste als Ausgabe. Geben Sie den Befehl aber mit dem Zusatz

```
cat list.txt | sort
```

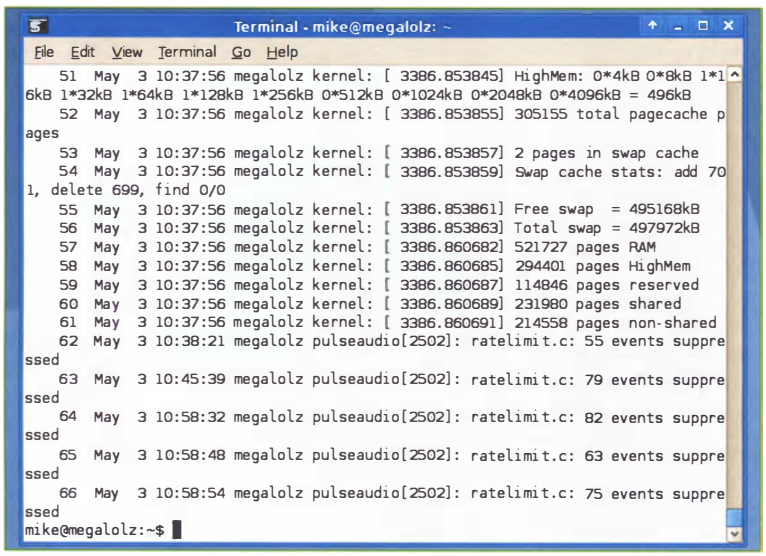
ein, dann sind die Zeilen in der Ausgabe alphabetisch sortiert. Sie haben also zwei Zeilen mit **ant**, zwei mit **bear** und eine Zeile mit **dolphin**. Wenn Sie noch die Option **-r** an den **sort**-Befehl anhängen, dann wird die Sortierreihenfolge umgekehrt, die Liste beginnt also mit **dolphin**.

Das ist alles schön und gut, aber es befinden sich Duplikate in der Liste, die Sie nicht brauchen können, weil sie die Bearbeitungszeit verlängern. Doch auch dafür gibt es eine Lösung in Form des **uniq**-Befehls und einer doppelten Pipe. Geben Sie einmal diesen Befehl ein:

```
cat list.txt | sort | uniq
```

Uniq sortiert nun die sich wiederholenden Zeilen im Textstrom aus und belässt nur die ursprüngliche. Entdeckt es zwei oder mehr Zeilen mit dem Wort **ant**, entfernt es also alle bis auf die erste. **Uniq** ist sehr leistungsstark und kommt mit einem ganzen Sack voller Optionen, mit denen Sie die Ausgabe weiter modifizieren können: Wenn Sie z.B. **uniq -u** eingeben, erscheinen nur Zeilen, von denen es kein Duplikat gibt, und mit **uniq -c** lassen Sie sich Zeilenzahlen anzeigen. Von unschätzbarem Nutzen ist **Uniq** für Sie besonders dann, wenn Sie mit großen Protokolldateien arbeiten, aus denen Sie jede Menge von irrelevantem Output herausfiltern möchten.

Weiter geht es mit der Formatierung von Text. Öffnen Sie die **list.txt**-Datei von vorhin und kopieren Sie den Inhalt so oft ein, dass der Text ungefähr 100 Zeilen lang ist. Speichern Sie, und



› Zählen Sie Ihre *PulseAudio*-Ausfälle in den Protokolldateien, indem Sie die Ausgabe mit der Pipe zum nl-Befehl leiten.

Text suchen mit dem mächtigen grep

Während **find** und **locate** bei Linux die Standardprogramme für das Suchen von Dateien sind, schaut **grep** ins Innere der Dateien, sodass Sie damit sogar bestimmte Worte und Ausdrücke suchen können. Hier ein einfaches Beispiel:

```
cat /var/log/messages | grep CPU
```

Dieser Befehl ruft alle Zeilen in der Datei **/var/log/messages** auf, die das Wort CPU enthalten. Beachten Sie, dass der Befehl standardmäßig zwischen Groß- und Kleinschreibung unterscheidet. Wenn Sie möchten, dass **grep** unabhängig von Groß- oder Kleinschreibung nach diesem Wort sucht, dann fügen Sie den **-i**-Parameter hinter dem **grep**-Befehl hinzu. Gelegentlich möchten Sie vielleicht eine Suche so durchführen, dass die gefundenen Zeilen aus der Ausgabe nicht angezeigt, sondern im Gegenteil herausgefiltert werden. Dazu können Sie den Parameter **-v** verwenden – dadurch werden alle Zeilen, die das Wort enthalten, ausgelassen.

grep arbeitet gut mit regulären Ausdrücken (siehe Seite 151) zusammen. In Letzteren gibt es ein paar Zeichen, die verwendet werden, um den Beginn und das Ende einer Zeile zu identifizieren. Erstellen Sie zu Demonstrationszwecken eine einfache Textdatei mit diesen drei Zeilen: **bird**, **badger**, **hamster**. Führen Sie dann diesen Befehl durch:

```
cat file.txt | grep -e ^b
```

grep wird durch diesen Befehl aufgefordert, eine Suche mittels regulärer Ausdrücke durchzuführen, wobei das ^-Zeichen sich auf den Beginn der Zeile bezieht. Es werden also nur die Zeilen ausgegeben, die mit **b** beginnen – **bird** und **badger**. Möchten Sie dagegen eine Suche starten, die sich auf die Zeilenenden bezieht, dann benutzen Sie das **$**-Zeichen:

```
cat file.txt | grep -e r$
```

Hier wird nun nach Zeilen gesucht, die mit dem Buchstaben **r** enden – das Ergebnis ist also **badger** und **hamster**. Um sehr komplizierte Suchparameter zu definieren, können Sie mehrere **grep**-Operationen nacheinander stellen und durch Pipes voneinander trennen.

Vor allem in älteren Unterlagen werden manchmal noch die Befehle **egrep** und **fgrep** erwähnt. Früher waren dies Varianten von **grep**, doch heute sind es nur noch Abkürzungen, um bestimmte Optionen des **grep**-Befehls zu spezifizieren. Weitere Informationen finden Sie via **man grep**.

führen Sie folgenden Befehl durch:

```
cat list.txt | fmt
```

Das **fmt**-Dienstprogramm formatiert Text in unterschiedlichen Formen und Stilen. Standardmäßig nimmt es sich unsere Liste – die Zeilen werden durch Newline-Zeichen voneinander getrennt – und formatiert das Ergebnis als einen normalen Textabschnitt, wobei es den Zeilenumbruch an die Breite des Terminalfensters anpasst. Den Zeilenumbruch können Sie mit dem Parameter **-w** bestimmen, indem Sie z.B. **cat list.txt | fmt -w 30** eingeben. Nun haben die Zeilen eine Höchstlänge von 30 Zeichen.

Wenn Sie Statistiken lieben, brauchen Sie eine Methode, mit der Sie die Zeilen in einem Textstrom zählen können. Das können Sie auf zweierlei Weise tun, entweder mit **nl** oder mit **wc**. Dabei ist **nl** eine direkte Methode, bei der einfach am Beginn eines Textstroms Zeilennummern eingefügt werden, z.B. so:

```
cat /var/log/messages | nl
```

Dadurch wird der Textinhalt von **/var/log/messages** ausgegeben, aber am Beginn jeder Zeile ist die Zeilennummer eingefügt. Wenn Sie die Ausgabe gar nicht sehen möchten, sondern nur die Nummern, dann verwenden Sie das **wc**-Dienstprogramm:

```
cat /var/log/messages | wc -l
```

wc steht für *word count* (Wortzahl). Führen Sie den Befehl ohne den Parameter **-l** durch, erhalten Sie ausführliche Informationen über Wortzahl, Zeilenanzahl und die Zeichenzahl des Textstroms.

So macht Formatieren Spaß

Das Abgleichen der Inhalte von Konfigurations- mit denen von Protokolldateien ist eine der Aufgaben, mit denen ein ausgebildeter Linux-Administrator sehr häufig beschäftigt ist. Wenn Sie ein erfahrener Programmierer sind, dann kennen Sie sich mit dem **diff**-Dienstprogramm gut aus und können damit Dateien abgleichen. **join** ist ein einfacheres Tool, das anzeigt, welche Zeilen sich in zwei Dateien entsprechen. Erstellen Sie eine Textdatei mit den Zeilen **bird**, **cat** und **dog**, und bezeichnen Sie die Datei als **file1**. Dann erstellen Sie **file2** mit den Zeilen **adder**, **cat** und **horse**. Führen Sie diesen Befehl durch:

```
join file1 file2
```

Das Wort **cat** wird ausgegeben, da es das einzige Wort ist, das in beiden Dateien steht. Wenn die Übereinstimmungen nicht nach Klein- und Großschreibung unterschieden werden sollen, dann fügen Sie den Parameter **-i** an den Befehl an.

Um Dateien zu teilen, verwendet man den Befehl **split**, der sich sowohl für Textinhalt als auch Binärdateien einsetzen lässt. Beim Textinhalt können Sie festlegen, in wie viele Zeilen Sie eine Datei teilen möchten, indem Sie den Parameter **-l** verwenden:

```
split -l 10 file.txt
```

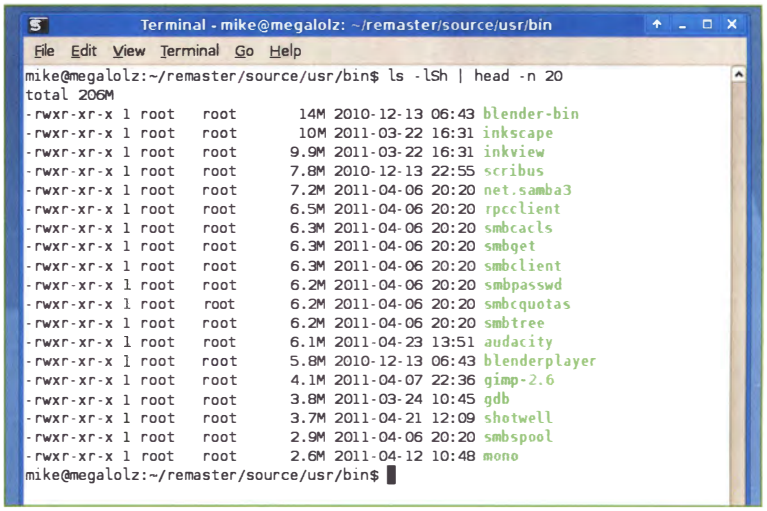
Die Datei **file.txt** wird in mehrere kleinere Dateien mit je 10 Zeilen geteilt, deren Dateinamen mit **xaa** beginnen, dann **xab**, **xac** usw. – wie viele Dateien entstehen, hängt von der Größe der Originaldatei ab. Die gleiche Operation können Sie auch mit Nicht-Textdateien durchführen, was sehr praktisch sein kann, wenn Sie eine Datei auf einen Datenträger transferieren müssen, für den die Datei zu groß ist. Beispielsweise passen auf einen FAT32-formatierten USB-Stick höchstens 4 GB. Eine 6-GB-Datei können Sie folgendermaßen teilen:

```
split -b 4096m largefile
```

Der Befehl teilt die Datei in zwei Dateien – die erste, **xaa**, ist 4 GB (4.096 MB) groß, die zweite, **xab**, enthält den Rest. Haben Sie die beiden Teile auf den Zielrechner transferiert, dann können Sie sie wieder zusammensetzen, indem Sie die zweite Datei mit diesem Befehl an die erste anhängen:

```
cat xab >> xaa
```

Jetzt enthält die Datei **xaa** alle ursprünglichen Daten, und Sie können ihr einen aussagekräftigeren Namen geben.



Sie möchten nur die ersten oder die letzten Zeilen der Ausgabe eines Befehls einsehen? Dabei helfen Ihnen die head- und tail-Befehle.

Weitere Utilities

Zum Schluss möchten wir noch ein paar andere Dienstprogramme erwähnen, die bei einer LPI-Prüfung vorkommen können. Wenn Sie die schieren Datenbytes einer Datei einsehen möchten, können Sie mit den Tools **hd** und **od** hexadezimale bzw. oktale Speicherausdrucke erstellen. In den entsprechenden Man-Pages können Sie die Unmengen von Parametern und Einstellungen nachschlagen, die für diese Befehle verfügbar sind.

Dann gibt es noch den **paste**-Befehl, der die Zeilen aus mehreren Dateien, nebeneinander und durch Tabs getrennt, auflistet. Mit dem Befehl **pr** können Sie Text für einen Ausdruck formatieren. Als Letztes sei noch **tr** erwähnt, ein Dienstprogramm, mit dem Sie einzelne Zeichen in einem Textstrom modifizieren oder entfernen. ■

Quick-Tipp

Der Umgang mit Textdateien, die Tabs enthalten, kann ziemlich schwierig sein. Die Lösung für das Problem ist der **expand**-Befehl. Dieser ändert Tabs in Leerstellen um, wodurch es einfacher ist, mit dem Text zu arbeiten. Um die Leerstellen wieder in Tabs zurückzuverwandeln, benutzen Sie den entsprechenden **unexpand**-Befehl.

Testen Sie Ihr Wissen!

Haben Sie sich die Lektionen eingehend durchgelesen und alle Befehle auf Ihrem eigenen Rechner selbst ausprobiert? Haben Sie die hier erläuterten Konzepte schon voll verinnerlicht? Dann wird es Zeit, dass Sie Ihr Wissen unter Beweis stellen!

1. Sie haben eine Datei namens **data.txt** und möchten die Ausgabe des **uname**-Befehls hinzufügen. Wie machen Sie das?
2. Wie zeigen Sie die Ausgabe von **df** an und leiten sie gleichzeitig in die Datei **myfile.txt**?
3. Die Datei **file.txt** enthält diese Zeile: **bird,badger,hamster**. Wie lösen Sie das zweite Wort heraus?
4. Sie haben eine Datei mit 500 Zeilen, die Sie in zwei Teile mit jeweils 250 Zeilen aufteilen möchten. Wie machen Sie das?
6. Und wie setzen Sie die beiden Teile wieder zu einer Datei zusammen?
7. In der Datei **file1.txt** möchten Sie das Wort **Windows** durchgehend durch **MikeOS** ersetzen. Wie gehen Sie vor?
8. Sortieren Sie **myfile.txt**, entfernen Sie alle Duplikate, und fügen Sie bei der Ausgabe eine Zeilennummerierung hinzu.

1 uname >> data.txt **2** df | myfile.txt **3** cat file.txt | cut -d"," -f 2 **4** split -l 250 file.txt **5** cat xab >> xaa **6** cat file1.txt | sed s/Windows/MikeOS/g > output.txt **7** cat myfile.txt | sort | uniq | nl

Lernen Sie Linux

MIT MIKE SAUNDERS

Lektion 7: Lernen Sie, wie Sie profimäßig mit Prozessen umgehen und wie Sie den berühmten minimalistischen Editor Vi einsetzen

Linux Professional Institute

Im vorletzten Teil unseres Lehrgangs kommen wir zu einigen Profi-Themen, denen Sie auf Ihrem Weg zum Linux-Systemadministrator begegnen könnten.

Wir starten mit einer Betrachtung von Prozessen und deren Anpassung. Es gibt nichts Schlimmeres als einen kaputten Prozess, der wichtige Dateien löscht und Sie hilflos zurücklässt. Deshalb sehen wir uns eine Lösung dieses Problems an. Wir befassen uns auch mit Dateisystemen, und zwar nicht wie bisher in Bezug auf deren Inhalt, sondern wie Sie Partitionen mit neuen Dateisystemen formatieren und Prüfungen durchführen, falls einmal etwas schief gehen sollte.

Danach geht es um den *Vi*-Editor, der fast jeder Linux-Distribution auf diesem Planeten beiliegt und dessen Bedienung zunächst schwierig sein kann, der aber toll ist, wenn man ihn gut beherrscht.

Teil 1: Prozesse verwalten

Stellen Sie sich vor, Sie bestellen im Lokal einen Tee und sehen, wie die Bedienung daraufhin die Kaffeemaschine anwirft. Dann würden Sie vielleicht schnell „Halt!“ rufen. Doch was müssen Sie tun, um bei einem Computerprogramm einen fehlgeleiteten Prozess abzubrechen? Die erste Möglichkeit ist es, STRG+C zu drücken. Testen Sie das mal mit einem Kommando, das eine sehr lange Ausgabe produziert, beispielsweise **ls –R /**, welches alle Ordner und deren Unterordner im Root-Verzeichnis auflistet. Während Tausende Zeichen Ihr Terminal-Fenster füllen, genügt das Drücken von STRG+C, um den Befehl abzubrechen. Er hört sofort auf zu arbeiten, es gibt nichts weiter, was Sie tun könnten. Dieses Wissen ist sehr wichtig, wenn Sie feststellen, dass Sie gerade etwas Verrücktes eingegeben haben und die Ausführung stoppen möchten, bevor Schaden angerichtet ist.

Es gibt allerdings noch eine Alternative zu diesem Befehl. Was, wenn Sie die Ausführung des Programms nur für später pausieren möchten? Nehmen wir an, Sie hätten gerade **man gcc** aufgerufen, um den Handbucheintrag für den GNU C-Compiler zu lesen. Sie haben etwas herumgescrollt und eine interessante Stelle in der Dokumentation entdeckt – Sie möchten im Terminal einige Dinge ausprobieren, ohne die Stelle in der Dokumentation zu verlieren. Mit STRG+Z verschwindet die Dokumentation im Hintergrund und die Kommandozeile kommt wieder in den Vordergrund. Hier können Sie alles Gewünschte ausprobieren, um nachher mit der Eingabe von **fg** wieder zur Dokumentation von gcc zurückzukehren, exakt zu der Stelle, an der Sie sie verlassen haben.

Es ist auch möglich, ein Programm unsichtbar zu starten. Sie rufen es in den Vordergrund, sobald Sie es brauchen. Sie machen das, indem Sie dem Programmaufruf ein kaufmännisches Und (**&**) anhängen, beispielsweise wie bei

```
man df &
```

Damit starten wir die Man-Page für **df** (Anzeigen des freien Laufwerksspeichers) und sehen eine Ausgabe auf dem Bildschirm:

```
[1] 3192
```

Die zweite Zahl ist die Prozess-ID (PID). Nun erledigen Sie

Prozesse priorisieren mit nice

Normalerweise hat irgendein Prozess auf dem System nicht mehr Anrecht auf Ressourcen als jeder andere. Wenn die Prozesse A und B gestartet werden und beide nach CPU-Leistung verlangen, wird der Kernel-Scheduler von Linux die CPU-Zeit gerecht zwischen ihnen aufteilen. Das ist jedoch nicht immer gewünscht, vor allem, wenn viele Prozesse im Hintergrund arbeiten. Sie könnten beispielsweise einen Cron-Job (eine regelmäßig ausgeführte Aufgabe) erstellt haben, der alte Archivdateien auf dem Desktop komprimiert: Wenn Sie gerade irgendetwas Wichtiges erledigen, möchten Sie ungern plötzlich 50 % der Leistung verlieren, nur weil der Cron-Job gerade seine Arbeit startet. Um dies zu vermeiden, gibt es ein System von Prioritätsstufen. Jedem Prozess ist ein **nice**-Wert zugewiesen, der angibt, wie priorisiert der Prozess vom Betriebssystem behandelt werden soll. Der Wert 19 steht für die niedrigste Priorität, -20 für die höchste und 0 für Standard. Wenn Sie ein Programm in der niedrigsten Prioritätsstufe starten möchten, benutzen Sie dazu den Befehl:

```
nice -n 19 Programmname
```

Dieser wird das Programm aufrufen, und es wird, sofern gerade nichts anders auf dem Computer passiert, mit einer normalen Geschwindigkeit ablaufen. Wenn das System jedoch während der Bearbeitung des Programms durch andere Prozesse beansprucht wird, wird es diese zuerst abarbeiten. Für **nice**-Werte über Null brauchen Sie Root-Rechte:

```
sudo nice n -10 Programmname
```

Dies ist für Mehrnutzersysteme gedacht, wo der Administrator seinen Aufgaben eine höhere Priorität zuweisen möchte. Sie können den nice-Wert eines Prozesses mit dem **renice**-Kommando ändern – weitere Informationen finden Sie auf der **renice**-Man-Page. Die aktuellen nice-Werte von Programmen lesen Sie mit **top** aus.

alles Notwendige und rufen, sobald Sie bereit sind, das Programm mit **fg** in den Vordergrund. Dieses System ist besonders nützlich, wenn Sie viele Programme gleichzeitig anwenden. Geben Sie als Beispiel Folgendes ein:

```
nano &
man df &
```

Hier haben wir zwei Programme im Hintergrund gestartet. Wenn wir **jobs** eingeben, bekommen wir eine Liste der Hintergrundprogramme mit einer vorangestellten Nummer und ihren Aufrufbefehlen. Einzelne Programme adressieren wir mit einer Nummer, beispielsweise wird **fg 1** den *Nano*-Editor starten, **fg 2** führt zur Man-Page von **df**.

Kommen wir nun zu Prozessen. Letztendlich sind Prozesse einzeln aufgeführte Instanzen von Programmen. Die meisten einfachen Programme bestehen aus einem Prozess, der das Programm selbst darstellt. Komplizierte Software-Suiten wie etwa eine Desktop-Umgebung starten viele Prozesse – Dateiüberwachungs-Daemons, Fenster-Manager und so weiter.

Die Eingabe von **ps** zeigt eine Liste der Prozesse an, die vom aktuellen Benutzer gestartet wurden. Wollen Sie alle laufenden Prozesse sehen, ist der Befehl **ps ax** notwendig. Da die Ausgabe sehr umfangreich sein wird, können Sie den Aufruf mit **less** kombinieren, damit die Ausgabe bildschirmweise erfolgt:

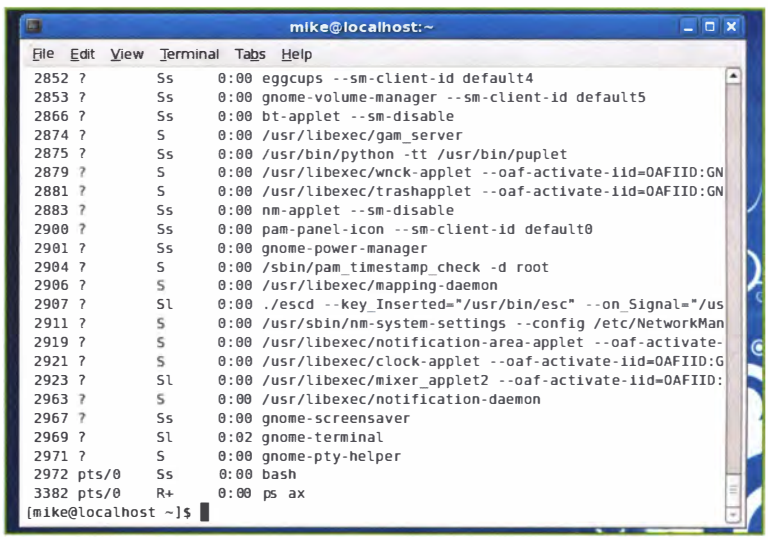
```
ps ax | less
```

Was genau Sie sehen, hängt von der exakten Aufmachung Ihres Linux-Systems und den zurzeit laufenden Programmen ab, aber hier als Beispiel mal eine Zeile:

```
2972 pts/0        Ss        0:00 bash
```

Die **2972** ist die Prozess-ID. Jeder Prozess besitzt eine einzigartige ID mit einer fortlaufenden Zählung ab 1, die für das Programm **/sbin/init** steht, das beim Systemstart den Kernel ausführt. Das nachfolgende **pts/0** zeigt an, von welchem virtuellen Terminal das Programm gestartet wurde. Falls Sie hier ein **?** sehen, ist es ein Prozess, der von außerhalb eines Terminals gestartet wurde, beispielsweise vom Kernel oder einem Boot-Skript. **Ss** sagt aus, dass der Prozess gerade schläft (also keine Aufgaben durchführt). Dann gibt es noch einen Timer, der zeigt, wie viel CPU-Zeit bis jetzt auf den Prozess entfallen ist, gefolgt vom Kommando, das benutzt wurde, um den Prozess zu starten.

Eine andere Methode, um eine Liste aller Prozesse zu bekommen, ist der Befehl **top**. Dieser sortiert die Prozesse nach CPU-Auslastung. Es ist ein interaktives Programm, das die Liste alle paar Sekunden aktualisiert und die rechenintensivsten



Das ist die Ausgabe von ps ax, das alle laufenden Prozesse auf dem System anzeigt.

Prozesse ganz oben anzeigt. Es bietet auch Überschriften für die Spalten, welche die PID, den User, der den Prozess gestartet hat, und Anderes anzeigen. Beachten Sie, dass es mehrere Spalten für den Speicherverbrauch gibt. Die wichtigste ist RES (resident), die anzeigt, wie viel realen Speicher ein Programm gerade nutzt. Die Taste Q beendet **top**.

Nehmen wir an, dass Sie gerade ein Programm ausführen und dieses plötzlich durchdreht, in einer Schleife steckt und so ihre CPU voll auslastet. Sie können es nicht mit STRG+C beenden, da Sie es womöglich über den Fenstermanager gestartet haben. Was können Sie noch machen? Die erste Möglichkeit ist, die PID des Prozesses mit einer der ersteren Methoden herauszufinden und danach folgenden Befehl einzugeben:

```
kill <pid>
```

Ersetzen Sie dabei **<pid>** mit der Nummer des störenden Prozesses. Das Kommando **kill** klingt allerdings martialischer, als es tatsächlich ist: In Wirklichkeit sendet es dem Programm ein freundliches „Würdest du dich bitte beenden?“, auf die der Prozess noch reagieren kann (beispielsweise, indem er die temporären Dateien aufräumt). Falls ein Programm einen eigenen **kill**-Signal-Handler nutzt und dieser aktuell zu ausgelastet ist, um auf das Signal zu reagieren, stecken Sie jedoch fest. Nun muss **kill** also doch etwas unfreundlicher werden. Tippen Sie:

```
kill -9 <pid>
```

Quick-Tipp

Die Tastenkombinationen zum Stoppen und Pausieren von Prozessen wie STRG+C und STRG+Z funktionieren in den meisten Fällen, aber nicht in allen. Programmen ist es erlaubt, diese Tastenkombinationen neu zu belegen um ihnen andere Funktionen zuzuweisen.

Dieser Modus von **kill** kümmert sich nicht darum, ob das Programm mit dem Befehl einverstanden ist – es wird umgehend beendet. Falls der Prozess gerade mitten beim Schreiben einer Datei ist, könnte das Ergebnis ziemlich verheerend sein. Deshalb sollte dieser Befehl nur mit Bedacht eingesetzt werden, wenn es keine andere Möglichkeit mehr gibt.

Manchmal haben Sie auch eine Anzahl von Prozessen mit dem gleichen Namen, über deren PID Sie sich gar nicht erst schlau machen wollen. In diesem Fall ist der **killall**-Befehl der richtige. Angenommen, Sie haben gerade ein topaktuelles *Apache*-Modul kompiliert, es in *Apache* eingefügt, und nun beginnt Ihr Webserver abzudrehen. Sie können *Apache* nicht mehr über seine normalen Skripte stoppen, und es laufen Unmengen von Prozessen mit **apache** im Namen. Probieren Sie in diesem in diesem Fall Folgendes aus:

```
killall -9 apache
```

Ein anderes nützliches, nicht destruktives Signal, das das Programm zu einem Neustart oder zum erneuten Einlesen der Konfigurationsdateien anweist, ist **SIGHUP**. Viele Programme ignorieren das Signal, es funktioniert aber wunderbar für viele Hintergrunddienste wie etwa Server:

```
killall -HUP sendmail
```

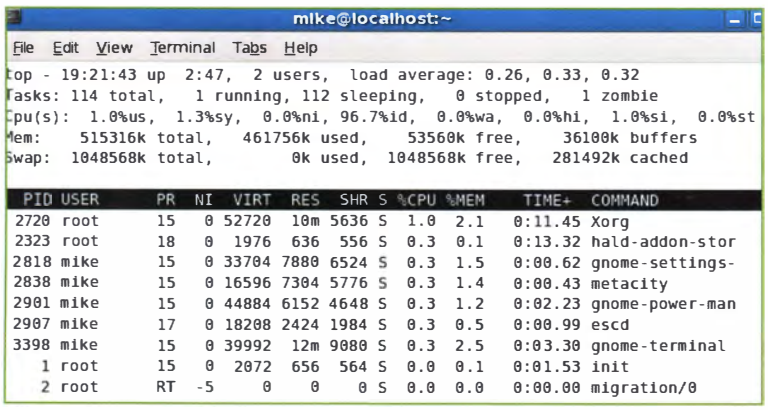
Alle **sendmail**-Prozesse werden hierbei angewiesen, für einen Moment zu pausieren und die Konfiguration neu einzulesen, um anschließend weiterzulaufen. Das ist sehr nützlich, wenn Sie eine kleine Änderung in einer Konfigurationsdatei vornehmen und dafür nicht das ganze Programm stoppen möchten. Sollten Sie aus irgendeinem Grund nicht wollen, dass das Signal beachtet wird, können Sie es mit dem **nohup**-Werkzeug unterbinden – wie genau, erfahren Sie in **man nohup**.

Quick-Tipp

Mit **uptime** können Sie sich einen schnellen Überblick darüber verschaffen, wie Ihr System arbeitet. Der Befehl zeigt die Zeit an, die seit dem letzten Reboot vergangen ist, wie viele Nutzer angemeldet sind und wie stark das System in letzter Zeit ausgelastet war.

Teil 2: Neue Dateisysteme anlegen

Mit dem nächsten Thema wird ein Systemadministrator nicht sehr häufig zu tun haben, aber damit auskennen sollte er sich doch: Dateisysteme und Partitionierung. In den meisten Fällen legen Sie die Partitionierung bei der Installation eines Computers fest, womit sich die Sache für Jahre oder zumindest für Monate erledigt hat. Grafische Werkzeuge wie etwa *GParted* werden von vielen Installern genutzt und machen den Vorgang zum Kinderspiel. Für Notfälle sollte man sich aber trotzdem mit den Kommandozeilenwerkzeugen auskennen.



Die Eingabe von top zeigt eine Liste aller Prozesse, sortiert nach ihrem CPU-Verbrauch.

Frischen wir unser Wissen etwas auf: Eine Festplatte ist normalerweise in mehrere Partitionen unterteilt. Sie könnten beispielsweise eine Partition für Windows nutzen und eine andere für Linux. Viele Linux-Installationen verteilen sich über mehrere Partitionen: Datenpartitionen wie / und **/home** sowie Swap-Partitionen für den virtuellen Speicher. Die Daten in diesen Partitionen sind auf eine bestimmte Art und Weise organisiert – formatiert. Früher nutzte Linux das **ext2**-Dateisystem, welches von **ext3** mit Journaling abgelöst wurde. Aktuell nutzen wir **ext4**. Andere Dateisysteme aus der Linux/Unix-Welt sind XFS und ReiserFS. Die meisten Windows-Computer nutzen NTFS, externe Speicher wie USB-Sticks sind oft mit FAT32 formatiert (unter Linux auch VFAT genannt). Das grundlegende Werkzeug zum

Die Integrität eines Dateisystems überprüfen

Moderne Linux-Dateisysteme wie **ext4** (der Standard aller modernen Installationen) sind robust und verlässlich. Im Falle eines Stromausfalles können Sie zwar keine Wunder vollbringen, allerdings sind sie in der Lage, das Dateisystem in einem nachvollziehbar konsistenten Zustand zu halten, sodass Sie nicht alle Ihre Daten verlieren. Nichts ist jedoch unkaputtbar, und wenn Sie ein größeres Problem mit Ihrer Hardware hatten, möchten Sie vielleicht überprüfen, ob die Festplatte noch in Ordnung ist. Folgendermaßen würde ein Administrator in diesem Falle vorgehen. Geben Sie zuerst **dmesg** ein, und überprüfen Sie, ob irgendetwas Seltsames in dem Log zu finden ist – Hinweise auf beschädigte Daten, kaputte Sektoren oder Ähnliches. Falls Sie irgendwas in dieser Richtung entdecken, sollten Sie sofort ein USB-Laufwerk anstöpseln und alle wichtigen Daten sichern. Sie können nämlich nicht wissen, ob sich das Laufwerk nicht schon bald endgültig verabschiedet. Verwenden Sie als Nächstes das Kommando **df- h**, um den freien Speicher auf dem Laufwerk zu überprüfen. Falls die Menge deutlich kleiner als erwartet ausfällt, könnte etwas nicht stimmen. Verwenden Sie **du -h** in den einzelnen Ordnern, um den Speicherverbrauch zu überprüfen. Der nächste Schritt ist das Durchführen einer Dateisystemüberprüfung. Starten Sie Ihre Distribution im Einzelbenutzermodus. Sobald Sie die Eingabeaufforderung erreichen, geben Sie Folgendes ein:

```
/sbin/fsck Partition
```

Passen Sie die **Partition** so an, dass sie auf Ihre Root-Partition, zum Beispiel **/dev/sda1** verweist. Sie können auch andere Partitionen überprüfen. **fsck** ist eigentlich ein Front-End für diverse Werkzeuge zum Dateisystem-Check. Auf den meisten Linux-Systemen läuft **/sbin/e2fsck**, das mit **ext2/3/4**-Partitionen umgehen kann. Falls während der Überprüfung Fehler auftauchen, wird **fsck** Sie fragen, wie es damit umgehen soll. Nach dem Überprüfungsvorgang können Sie **/sbin/dumpe2fs**, gefolgt von der Gerätebezeichnung eingeben, was hilfreich sein kann, wenn Sie sich mit Ihrem Problem an ein Onlineforum wenden möchten. Sie werden feststellen, dass viele Linux-Distributionen **fsck** automatisch nach 30 Startvorgängen, nach 100 Tagen oder einer Kombination aus beidem durchführen. Sie können diese Einstellungen mit **tune2fs** und den Optionen **-c** und **-C** anpassen. Es gibt auch Einstellungsmöglichkeiten dafür, wie der Kernel mit Fehlern umgehen soll, falls er welche feststellt. Auf der Man-Page finden Sie reichlich Informationen zu den genannten Optionen.

Formatieren von Laufwerken aus der Kommandozeile ist *fdisk*. Teilen Sie dem Tool den Gerätenamen auf folgende Weise mit:

```
/sbin/fdisk /dev/sda
```

Sie müssen hierzu Root-Rechte besitzen. Falls Sie ihre Laufwerksbezeichnung nicht kennen, sehen Sie sich die Ausgabe von **dmesg** an. Achten Sie darauf, dass Sie nur **/dev/sda** angeben, nicht **/dev/sda1** – die Zahl gibt die spezifische Partition an, wir interessieren uns jedoch für die komplette Platte. In den meisten Fällen ist **sda** das erste, **sdb** das zweite Laufwerk usw.

fdisk ist ein sehr spartanisches Programm. Es gibt keine Menüs oder Assistenten, die Vorgänge automatisieren. Die Eingabe von **p** zeigt eine Liste aller Partitionen an, **m** öffnet die Hilfe. Dort finden heraus, welche Kommandos Partitionen löschen, erstellen und so weiter. Alle Änderungen, die Sie ausführen, werden erst mit **w** auf den Datenträger geschrieben.

Manche Distributionen legen *cfdisk* bei, eine curses-basierte Version von *fdisk*, die die Arbeit um einiges erleichtert: Es gibt einfache Menüs, und Sie können mithilfe der Cursortasten navigieren.

Nachdem Sie eine Partition erstellt haben, müssen Sie sie formatieren. An dieser Stelle kommt *mkfs* ins Spiel. Geben Sie **/sbin/mkfs** ein und betätigen Sie die TAB-Taste, um eine Auswahl der Möglichkeiten zu sehen, darunter **mkfs.ext3** (für die meisten Linux-Partitionen) und **mkfs.vfat** (für FAT32-Partitionen). Um eine Partition zu formatieren, müssen sie nur das Gerät angeben:

```
/sbin/mkfs.ext3 /dev/sda2
```

Für swap-Partitionen benutzen Sie das **mkswap**-Kommando. Anschließend können Sie den Swap-Speicher mit **swapon** und **swapoff** ein- oder ausschalten.

Quick-Tipp

Nutzen Sie **free -m**, um einen Hinweis darauf zu bekommen, wie viel Speicher frei ist. Der Befehl zeigt Statistiken in Megabytes. Die erste Zeile könnte Sie vielleicht schockieren und Sie glauben machen, es wäre kaum noch Speicher frei. Die dortige Zahl beinhaltet jedoch auch Platten-Zwischenspeicher. Entscheidend ist vielmehr die zweite Zeile, **-/+ buffers/cache**.

Teil 3: Der Vi-Editor

Zuletzt nehmen wir noch den Editor *Vi* unter die Lupe. Zu der Zeit, als *Vi* für Unix-Betriebssysteme entwickelt wurde, nutzten manche Leute noch Fernschreiber. Das Konzept eines Vollbild-Editors für Text war damals noch Neuland, und viele waren es noch gewohnt, Text Zeile für Zeile zu verarbeiten. *Vi* ist immer noch sehr spartanisch. Aufgrund seiner niedrigen Systemanforderungen ist er aber auf nahezu jedem Unix-Rechner installiert. Unter Linux nutzen die meisten Distributionen mit der Weiterentwicklung *Vim* (*Vi Improved*) eine wesentlich mächtigere Version des Editors.

Vi benutzen

Der Befehl **vi filename.txt** startet den Editor. Bevor Sie irgendeine Taste drücken, müssen Sie wissen, dass *Vi* in zwei Modi arbeitet, nämlich Normal (für Kommandos) und Insert (für Textbearbeitung). Das unterscheidet ihn von fast allen anderen Texteditoren, wo man einfach so drauflostippen kann. In *Vi* drücken Sie **i**, um Text an der aktuellen Position einzufügen. Sobald Sie fertig sind, erreichen Sie mit ESC wieder den Normal-Modus zur Kommandoeingabe.

Es gibt viele Kommandos, und wenn Sie ein *Vi*-Guru werden möchten, gibt es auch reichlich Literatur dazu. Hier aber erst mal etwas Grundlagenwissen: Wenn Sie im Normal-Modus **dd** eingeben, wird eine Zeile gelöscht, **yy** kopiert eine Zeile in die Zwischenablage, eingefügt wird sie mit **p**. Es gibt einige Operationen, die zuvor das Einfügen eines Doppelpunktes erfordern. Zum Beispiel schreibt **:w** die Datei auf den Datenträger, während **:q** den Editor beendet. Sie können sogar Aktionen kombinieren, wie Schreiben und Speichern mit **:wq**.

Viele Menschen finden *Vi* und *Vim* extrem unkomfortabel zum Arbeiten und bevorzugen Editoren ohne getrennte Modi wie *Emacs* oder *Nano*. Andere lieben den Minimalismus von *Vi* und hassen den STRG-Tastenwahnsinn der anderen beiden Editoren. Wie auch immer man persönlich dazu steht, kennen sollte man *Vi* als tüchtiger Administrator auf alle Fälle. ■

```
                VIM - Vi IMproved

                 version 7.0.237
              by Bram Moolenaar et al.
          Modified by <bugzilla@redhat.com>
     Vim is open source and freely distributable

            Help poor children in Uganda!
type    :help iccf<Enter>       for information

type    :q<Enter>               to exit
type    :help<Enter>  or  <F1>  for on-line help
type    :help version7<Enter>   for version info
```

› *Vim* zeigt im Gegensatz zu *Vi* einen hilfreichen Infotext an, wenn Sie das Programm ohne Dateinamen starten.

Testen Sie Ihr Wissen!

Auch zu dieser Lektion haben wir wieder ein paar Fragen für Sie. Versuchen Sie zunächst, die Fragen ohne nachzuschauen zu beantworten, um sich für die Situation einer Prüfung im Rahmen der LPI-Zertifizierung vorzubereiten.

1. Wie würden Sie das Kommando **exim -q** im Hintergrund ausführen?
2. Sie haben mehrere Programme im Hintergrund laufen. Wie zeigen Sie eine Liste von ihnen an?
3. Wie erstellen Sie eine Liste von Prozessen?
4. *Exim* ist durchgedreht, weshalb Sie alle Instanzen des Programms beenden müssen. Wie lautet das Kommando dafür?
5. Sie haben gerade die Partition **/dev/sda2** angelegt und möchten sie mit FAT32 formatieren. Wie gehen Sie dabei vor?
6. Zeigen Sie eine Möglichkeit, **myprog** mit der niedrigsten Priorität zu starten.

```
1 exim -q & 2 jobs 3 ps ax 4 killall -9 exim 5 /sbin/mkfs.vfat /dev/sda2 6 nice -n 19 myprog
```

Lernen Sie Linux

MIT MIKE SAUNDERS

Lektion 8: Im letzten Teil unserer Linux-Schulung beschäftigen wir uns mit der Erstellung von Links, der Rechteverwaltung und dem Festlegen von Speicherkontingenten.

Mike

Linux Professional Institute

Wir sind am Ende dieser Serie angekommen und haben bis hierher viel durchgenommen. Hoffentlich sind Sie zufrieden mit Ihrem Lernerfolg. Ob Sie ein fortgeschrittener Linux-Benutzer sind, der mit dem neu erworbenen Wissen eine neue Anstellung finden möchte, oder normaler Heimanwender, der nur etwas mehr über sein Betriebssystem erfahren wollte – wir wünschen Ihnen viel Erfolg und weiterhin viel Spaß mit Linux!

Es gibt nur noch wenige Themen, die wir noch nicht abgedeckt haben, denen wir uns aber jetzt widmen werden. Wir sehen uns Verknüpfungen im Dateisystem an, wie sich Berechtigungen auf die Sicherheit des Systems auswirken und wie Sie den Speicherplatzverbrauch mithilfe von Speicherkontingenten begrenzen.

Teil 1: Verknüpfungen erstellen

Jeder gute Systemadministrator versucht, doppelten Ressourcenverbrauch zu vermeiden. Nicht nur, weil das Verschwendung ist, es stiftet auch Verwirrung. Ein Beispiel: Eine Datei soll an zwei verschiedenen Orten im Dateisystem sichtbar sein. Das könnte eine Textdatei sein, die Sie aus ihrem Home-Verzeichnis heraus editieren möchten, die allerdings auch für Nutzer Ihres Webservers unter **/var/www/textfiles/** sichtbar sein muss.

Die einfachste Lösung für das Problem wäre es, die Datei regelmäßig vom Home-Verzeichnis nach **/var/www/textfiles/** zu kopieren. Das erfüllt den Zweck, sorgt aber für zwei Probleme:

» Wenn Sie das Kopieren der Datei vergessen, entwickeln sich die beiden Dateien nicht mehr synchron.

» Was, wenn ein anderer Administrator die Datei ändern möchte und stattdessen jene unter **/var/www/textfiles/** bearbeitet?

Wir umgehen dieses Problem mit symbolischen Verknüpfungen (Symlinks). Dabei handelt es sich um extrem kleine Dateien, die kaum mehr als einen Verweis auf eine andere Datei im Dateisystem enthalten – sehr ähnlich den Verknüpfungen in der Windows-Welt. Symbolische Verknüpfungen existieren für sich, sodass Sie sie einfach löschen können, ohne die Dateien zu beeinflussen, auf die sie zeigen. Erstellen wir mal einen Beispiel-Link. Wir beginnen damit, dass wir eine leere Datei anlegen:

```
touch myfile
```

Das erstellt eine leere, 0 Byte große Datei für Demonstrationszwecke, Sie können symbolische Verknüpfungen aber im Prinzip auf jede Datei zeigen lassen. Nun erstellen wir einen Link zu dieser Datei. Dazu nutzen wir folgendes Kommando:

```
ln -s myfile mylink
```

Die Reihenfolge der Kommandos ist wichtig: Wir haben den Parameter **-s**, der festlegt, dass es sich hier um einen Symlink handelt. Darauf folgen die Angabe zur Zieldatei und der Name der Linkdatei selbst. Wir arbeiten hier innerhalb desselben

Dateien finden

Es gibt zwei verschiedene Wege, Dateien in einem Linux-Dateisystem aufzuspüren: **find** und **locate**. Sie funktionieren allerdings unterschiedlich. Bei **find** handelt es sich um ein kleines Werkzeug, welches das Dateisystem durchgeht und jeden Dateipfad ausspuckt, der zu Ihrer Suchangabe passt. Geben Sie als Beispiel im Root-Verzeichnis (/) folgenden Befehl ein:

```
find . -name "linux"
```

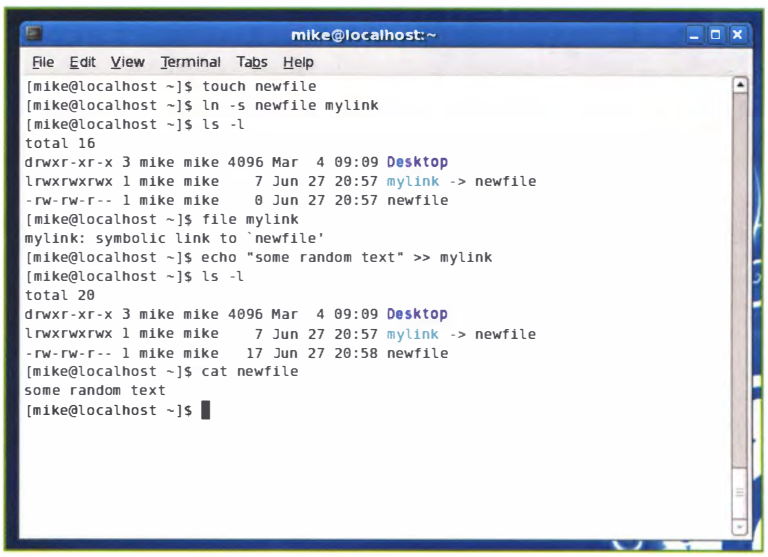
Dieser wird das aktuelle Verzeichnis samt allen Unterverzeichnissen nach Dateien durchsuchen, die das Wort „linux" enthalten. **find** hat eine Vielzahl von Optionen, die Sie der Man-Page entnehmen (**man find**). Dazu gehört das Suchen nach speziellen Dateitypen. Allerdings hat das Tool einen erheblichen Makel: Es arbeitet langsam. Sehr langsam. Vor allem auf Systemen, wo es Tausende von Dateien gibt. Die Lösung für dieses Problem heißt **locate**. Während sich **find** von Datei zu Datei durch das Dateisystem hangelt, konsultiert **locate** eine zuvor angelegte Datenbank. Die Suche arbeitet darum blitzschnell:

```
locate linux
```

locate hat auch einen Nachteil: Wenn Änderungen am Dateisystem gemacht werden, kann die **locate**-Datenbank veralten, sodass die Ergebnisse der Suche nicht zwingend richtig sein müssen. Die Datenbank lässt sich mit dem **updatedb**-Kommando aktualisieren. Normalerweise gibt es einen Cron-Job (geplante Aufgabe), der das täglich erledigt. Werfen Sie einen Blick in **/etc/cron.daily**, und suchen Sie eine Datei namens **locate**, **mlocate** oder **updatedb**. Im Inhalt der Datei erkennen Sie, dass die Datei das Kommando zum Update der Datenbank durchführt. Dieses liegt meist in **/var/lib/mlocate**. Die Einstellungen befinden sich dagegen unter **/etc/updatedb.conf**. Bedenken Sie außerdem, dass Sie bestimmte Dateisysteme und Einhängepunkte aus der Indexierung durch die Datenbank ausklammern können. So weichen Sie beispielsweise dem schmerzhaft langsamen Indexieren von DVDs aus.

Verzeichnisses, aber Sie können auch komplette Pfade zu den Dateien angeben. Auch Ordner funktionieren als Ziel. Wenn Sie **ls -l --color** eingeben, weist die Verknüpfung eine andere Farbe auf (siehe Abbildung rechts). Diese ausführliche Auflistung zeigt das Ziel der Verknüpfung nach dem **->**.

So weit, so gut. Aber wie funktioniert das Ganze in der Praxis? Nun, editieren Sie mal die Datei **mylink** – öffnen Sie sie in *Nano*, *Vim* oder *gEditor*, geben Sie etwas Text ein, und speichern Sie sie. Zurück in der Eingabeaufforderung geben Sie **ls -l** und **cat myfile** ein und werden feststellen, dass, obwohl Sie **mylink** editiert haben, alle Änderungen in **myfile** stattgefunden haben. Programme kümmern sich nicht um Links – sie suchen sich die Datei, auf welche die Verknüpfung verweist, und bearbeiten jene.



› Eine symbolische Verknüpfung in Aktion – wenn wir ihrem Inhalt Text anhängen, tun wir das in Wirklichkeit bei der Originaldatei.

Katastrophen vermeiden

Meist gibt es Sicherheitsvorkehrungen, um ein komplettes Desaster zu verhindern. Wenn Sie **rm mylink** eingeben, wird der symbolische Link gelöscht, nicht die Datei, auf die er zeigt. Andernfalls wäre es unmöglich, eine symbolische Verknüpfung zu löschen.

Aber es gibt auch einen anderen möglichen Fall: Gehen Sie die vorherige Abfolge nochmals durch, löschen Sie dieses Mal aber **myfile** statt **mylink**. Nun ist die Datei, auf die **mylink** verweist, nicht mehr existent, der Link ist kaputt. Wenn Sie **ls -l --color** eingeben, wird der Link mit einer warnenden roten Farbe dargestellt. Der Befehl **file mylink** informiert Sie, dass die Verknüpfung kaputt ist.

> „Symlinks bieten viele Möglichkeiten zur Flexibilisierung und Anpassung."

Symlinks bieten viele Möglichkeiten zur Flexibilisierung und Anpassung. Das erkennen Sie beispielsweise, wenn Sie einen Blick in **/etc/alternatives** werfen (sofern auf Ihrer Distribution vorhanden). Das Verzeichnis ist voll von symbolischen Verknüpfungen und erlaubt es Ihnen, verschiedene Versionen einer Software gleichzeitig auf dem System zu behalten. Es gibt beispielsweise verschiedene Versionen von Java, mit einem Symlink zeigt das **java**-Kommando immer genau auf die Version, die Sie verwenden möchten.

Auch wenn Sie eine experimentelle Version von *GCC* installiert haben: Sie verlinken **/usr/bin/gcc** symbolisch (symlinken) zur experimentellen Version, kompilieren die Dinge, die Sie kompilieren möchten, und symlinken später wieder die ursprüngliche Version.

Es gibt noch eine andere Art von Verknüpfung, die zudem noch wesentlich mächtiger ist: den Harten Link oder Hardlink. Wo die symbolische Verknüpfung den Dateinamen ihres Ziels beinhaltet, ist der Hardlink ein eigener Eintrag im Dateisystem. Um dies besser zu verstehen, vergegenwärtigen Sie sich, wie ein Dateisystem funktioniert: Es ist voll mit Daten, und am Anfang liegt eine Tabelle, die beschreibt, welche Datenstücke zu welcher Datei gehören. Stellen Sie sich nun vor, dass **myfile** eine Textdatei ist, deren Datenbereich an Position 63813 auf der Festplatte beginnt. Wenn Sie einen Symlink auf die Datei **myfile** erstellen, wird eine neue Datei angelegt, die eine Verknüpfung enthält. Diese sagt nichts weiter als „Hallo, eigentlich verweise ich auf die Datei **myfile**, also sieh dir die mal an." Wenn Sie einen Hardlink erstellen, werden hingegen keine neuen Dateien angelegt. Stattdessen wird die Verzeichnistabelle des Dateisystems aktualisiert, damit **mylink** »

» ebenfalls auf Position 63813 im Dateisystem verweist. Es gibt keine zwei getrennten Dateien, nur zwei Dateinamen, die auf den identischen Platz im Dateisystem verweisen.

Zusammenfassend: Symbolische Links zeigen auf andere Dateien, Hardlinks zeigen auf Datenbereiche in Dateisystemen. In der Praxis bedeutet das, dass sie anders als Symlinks funktionieren. Legen Sie beispielsweise eine Textdatei **foo** mit etwas Text darin an, und erstellen Sie einen darauf verweisenden Hardlink mit

```
ln foo bar
```

(beachten Sie, dass wir für einen Hardlink den Parameter **-s** weglassen). Geben Sie nun **rm foo** ein, um die Originaldatei zu löschen, und überprüfen Sie das Ergebnis mit **ls -l**. Sie stellen fest, dass **bar** immer noch in Ordnung zu sein scheint, auch **cat bar** zeigt, dass die Textinhalte immer noch vorhanden sind. Wie kann das sein? Nun, als Sie **foo** mit **rm** gelöscht haben, haben Sie nur den Eintrag aus der Verzeichnistabelle des Dateisystems gelöscht, nicht die Datei an sich. Es zeigt immer noch ein anderer Tabelleneintrag auf die Datei, nämlich **bar**.

Hardlinks haben Beschränkungen (sie dürfen nicht auf andere Dateisysteme verweisen) und werden selten genutzt, weshalb Sie sich auch nicht das Gehirn mit den technischen Details zu zermartern brauchen. Es ist aber sinnvoll, zu verstehen, was dabei passiert. Symbolische Verknüpfungen sind bei Weitem verbreiteter in Linux-Installationen. Wenn Sie sie zu erstellen in der Lage sind, ersparen Sie sich Einiges an Duplikaten auf Ihrem System. Sie werden vielen davon begegnen, zahlreiche Pakete legen sie selbstständig an.

Quick-Tipp

Es ist denkbar, zwei symbolische Verknüpfungen anzulegen, die wechselseitig aufeinander verweisen. Das klingt vermutlich so, als wolle man etwas durch Null teilen. Und Sie sind zu Recht vorsichtig, da das eine unendliche Schleife auslösen könnte, die Ihre CPU dahinschmelzen lässt. Linux ist hierbei aber in der Regel wachsam und wird Sie mit einer Meldung wie „Zu viele Ebenen mit einer symbolischen Verknüpfung" warnen.

Teil 2: Rechteverwaltung

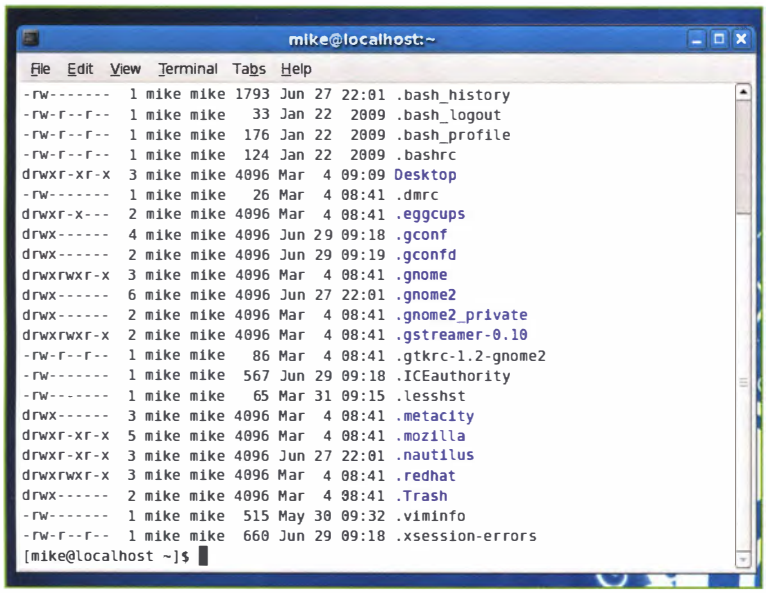
Rechte sind ein essenzieller Bestandteil eines jeden aktuellen Betriebssystems. Es wäre verrückt, wenn jeder Nutzer sämtliche Dateien auf dem System öffnen, ändern und löschen könnte. Sogar auf einem System, das nur ein einzelner Nutzer verwendet – so wie viele Desktops – ist es notwendig, ein funktionierendes System der Rechtevergabe zu haben. Dadurch werden viele Missgeschicke verhindert. Da ein normaler Nutzer keine Dateien in **/etc** oder **/boot** bearbeiten kann, zerstören seine Vertipper in der Kommandozeile auch keine für den Systemstart kritischen Dateien und machen damit das System unbenutzbar.

Daneben gibt es noch den Aspekt der Sicherheit. Da die meisten User darauf beschränkt sind, Dateien aus dem Home-Verzeichnis zu bearbeiten, löscht ein unerfahrener Benutzer mit einem heruntergeladenen Malware-Skript nur eigenen Dateien. Das kann unter Umständen natürlich sehr schlimm sein, doch immerhin kann der Schädling nicht die fundamentale Funktionsweise des Betriebssystems ändern.

Berechtigungen sind unter Linux, wie bei vielen unixbasierten Betriebssystemen, in drei Gruppen unterteilt:

» Lesezugriff
» Schreibzugriff
» Ausführberechtigung

Diese können Sie unabhängig voneinander verteilen. Für einen normalen Nutzer sind die Dateien in **/ect** beispielsweise nur lesbar, nicht aber veränderbar. Die Dateien in **/bin** und **/usr/bin** sind normalerweise lesbar und ausführbar, aber wiederum nicht beschreibbar. Als Randbemerkung: Verzeichnisse müssen sowohl lesbar als auch ausführbar sein, damit Sie darauf zugreifen können.



› Die Eingabe von ls -la zeigt alle aktuellen Dateien im aktuellen Verzeichnis, zusammen mit den Berechtigungen ganz links in der Zeile.

Gruppen

Berechtigungen sind in Gruppen organisiert. Es gibt den Eigentümer der Datei, die Gruppe, zu welcher der Eigentümer gehört, sowie Drittnutzer. Lassen Sie uns diese nacheinander behandeln. Geben Sie in einem Terminal folgende Befehle ein:

```
touch foo
ls -l foo
```

Dies erstellt eine neue Datei namens **foo** und zeigt die

Speicherplatz mittels Kontingenten beschränken

Auf einem Multiuser-System kann es schon mal vorkommen, dass Sie den Speicherplatzverbrauch begrenzen möchten. Ein Schnellschuss für die Lösung dieses Problems wäre es, die Größe der **/home**-Partition zu verkleinern, was allerdings sehr unflexibel ist. Eine bessere Herangehensweise ist die Verwendung von Speicherkontingenten. Die Einrichtung nutzt diverse Prozesse und unterscheidet sich von Distribution zu Distribution. Anstatt also jeden Schritt lang und breit zu beschreiben, konzentrieren wir uns hier auf die Essenz. Die spezifischen Informationen zur Einrichtung finden Sie im Internet (falls die Dokumentation Ihrer Distribution sehr gründlich ist, wie die von Gentoo, auch dort). Kontingente beschränken den Speicherverbrauch für Nutzer oder für Gruppen. Sie werden **/etc/fstab** editieren müssen – die Datei, in der festgelegt wird, wie die Dateisysteme gemountet werden – um die Optionen **usrquota** und **grpquota** für das betreffende Dateisystem einzufügen. Mounten Sie das Dateisystem im Einzelnutzermodus, legen Sie die Einstellungsdateien **aquota.user** und **aquota.group** an, und erstellen Sie einen Cron-Job für **quotacheck**, um regelmäßig zu überprüfen, ob ein Nutzer sein Kontingent überschritten hat. Mit dem Kommando **edquota** legen Sie die Kapazitäten für die Nutzer fest. Die Befehle **quotaon** und **quotaoff** (de-)aktiveren die Kontingente. **requota** meldet den Speicherverbrauch.

standardmäßig für sie gesetzten Berechtigungen an. Der hier relevante Teil ist **-rw-r--r--** am Anfang der Zeile. Lassen Sie uns das in seine Bestandteile aufbrechen:

1 **-** Der erste Bindestrich bezeichnet den Dateityp: **d** steht für ein Verzeichnis, **l** für einen Link und **-** für eine einfache Datei.

2 **rw-** Das erste Bündel aus drei Zeichen zeigt die Rechte des Eigentümers der Datei an. **rw** steht für Lese- und Schreibrechte, der Strich steht dafür, dass die Datei nicht ausführbar ist. (**rwx** würde bedeuten, dass die Datei auch ausführbar wäre.)

3 **r--** Das nächste Trio beschreibt die Rechte der Gruppe. In diesem Fall ist die Datei für die Gruppe lesbar, aber weder editier- noch ausführbar. Linux-Konten können in Gruppen zusammengefasst werden, um über diese die Rechte der zugehörigen Nutzer zu verwalten. Falls beispielsweise 50 Personen an Ihrem Rechner arbeiten, aber nur 20 davon Zugriff auf einen bestimmten Ordner haben sollen, können Sie diese 20 Personen zu einer Gruppe zusammenfassen und ihr Schreibrechte für den Ordner gewähren.

4 **r--** Die letzten drei Zeichen gelten für alle anderen Nutzer auf dem System. Die Datei ist nur lesbar, nicht aber änderbar oder gar ausführbar.

Manche Kombinationen aus bestimmten Rechten muten seltsam an, man weiß aber nie, wann sie gebraucht werden könnten. Wenn Sie ein Programm als ausführbar, aber nicht lesbar markieren, können es die betreffenden Nutzer zwar starten, aber nicht im Code herumschnüffeln, zum Beispiel mit dem **strings**-Tool.

Rechte modifizieren

Um die Rechte anzupassen, verwenden wir das **chmod**-Werkzeug. Das funktioniert so, dass Sie einen Buchstaben für den Nutzertyp und ein Plus oder Minus für die zu ändernde Berechtigung angeben. Die Nutzertypen im Einzelnen sind:

- **u** = der Eigentümer
- **g** = die Benutzer in der Gruppe
- **o** = die anderen Nutzer
- **a** = alle/jedermann

Soll beispielsweise nur der Eigentümer eine Datei ausführen können, ist folgender Befehl nötig:

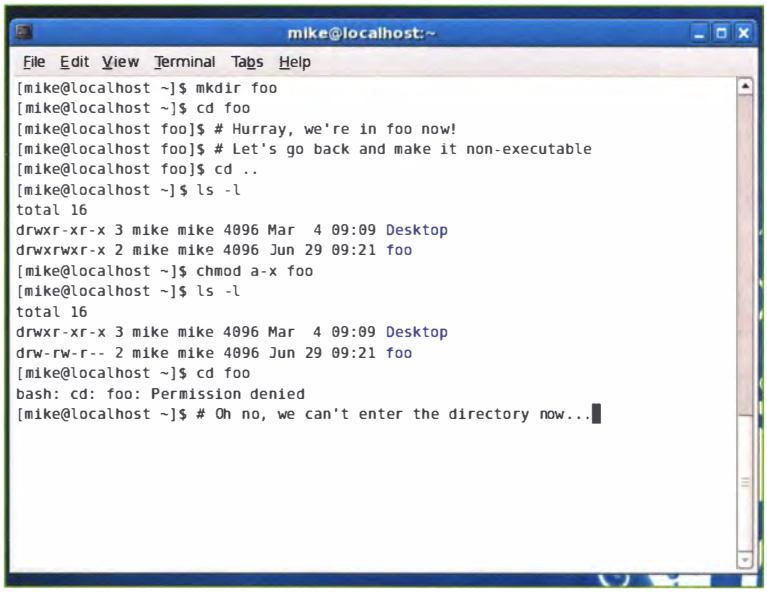
```
chmod u+x meinskript
```

Um die Datei für jeden (außer dem Root) unlesbar zu machen:

```
chmod a-r meinskript
```

Sie können **chmod** auch rekursiv auf Verzeichnisse anwenden, indem Sie den Schalter **-R** nutzen. Weitere Informationen zu **chmod** finden Sie unter **man chmod**.

So viel zum Rechtemanagement. Wie aber verwalten wir Eigentümer und Gruppen? Der Befehl **chown** (von „change ownership“) akzeptiert einen Nutzer und eine Gruppe als Argument,



› Falls Sie einmal Probleme beim Öffnen eines Verzeichnisses haben sollten, stellen Sie sicher, dass das Verzeichnis als ausführbar markiert ist.

beide getrennt durch einen Doppelpunkt. Als Beispiel:

```
chown stefan:vboxnutzer meinedatei
```

Wenn Sie sich die Datei nun mit **ls -l** ansehen, werden Sie feststellen, dass ihr Eigentümer **stefan** zur Gruppe **vboxnutzer** gehört. Werfen Sie einen Blick in **/etc/group**, um eine Liste der verfügbaren Nutzer zu sehen. Wie Sie eine Gruppe anlegen, erfahren Sie auf der Man-Page von **usermod** (speziell die Schalter **-a** und **-G**). Während **chown** dazu dient, den Eigentümer zu ändern, gibt es ein eigenständiges **chgrp**-Tool zum Anpassen der Gruppe.

Nimm deine Maske ab

Eine ziemlich fortgeschrittene Funktion, die Sie kennen sollten, ist **umask**. Dabei handelt es sich um eine Einstellung der Shell, die festlegt, mit welchen Rechten der Standardnutzer darin ausgestattet ist.

Die Standardeinstellung der meisten Linux-Systeme ist, dass Dateien für den Ersteller les- und editierbar sind. Andere Nutzer auf dem System oder auch aus derselben Gruppe können sie hingegen nur lesen. Das kann mit dem **umask**-Befehl angepasst werden. Geben Sie folgendes Kommando in die Eingabeaufforderung ein:

```
umask a+rw
```

Jetzt legen wir mit **touch blah** eine Datei an. Mit **ls -l blah** sehen Sie, dass die Datei für jeden les- und änderbar ist. Sie können das nutzen, um die Benutzerkonten über die *Bash*-Konfigurationsdateien (**~/.bashrc**) oder systemweit anzupassen. ■

Quick-Tipp

Nutzen Sie **whlch**, um die Binary eines Kommandos zu finden, zum Beispiel **whlch ls**. Es gibt auch eine Variante **wherels**, die anzeigt, wo die Man-Page gefunden werden kann.

Linux – machen Sie was draus!

Herzlichen Glückwunsch! Sie haben unseren Linux-Lehrgang bis zum Ende durchgearbeitet und wissen jetzt hoffentlich gut über die grundlegenden Aspekte des Betriebssystems Bescheid. Darüber hinaus haben Sie nun von vielen weiterführenden Themen, auch wenn wir sie zum Teil nur kurz anreißen konnten, zumindest schon einmal gehört – dies wird Ihnen helfen, wenn Sie sich weiter in die Materie vertiefen.

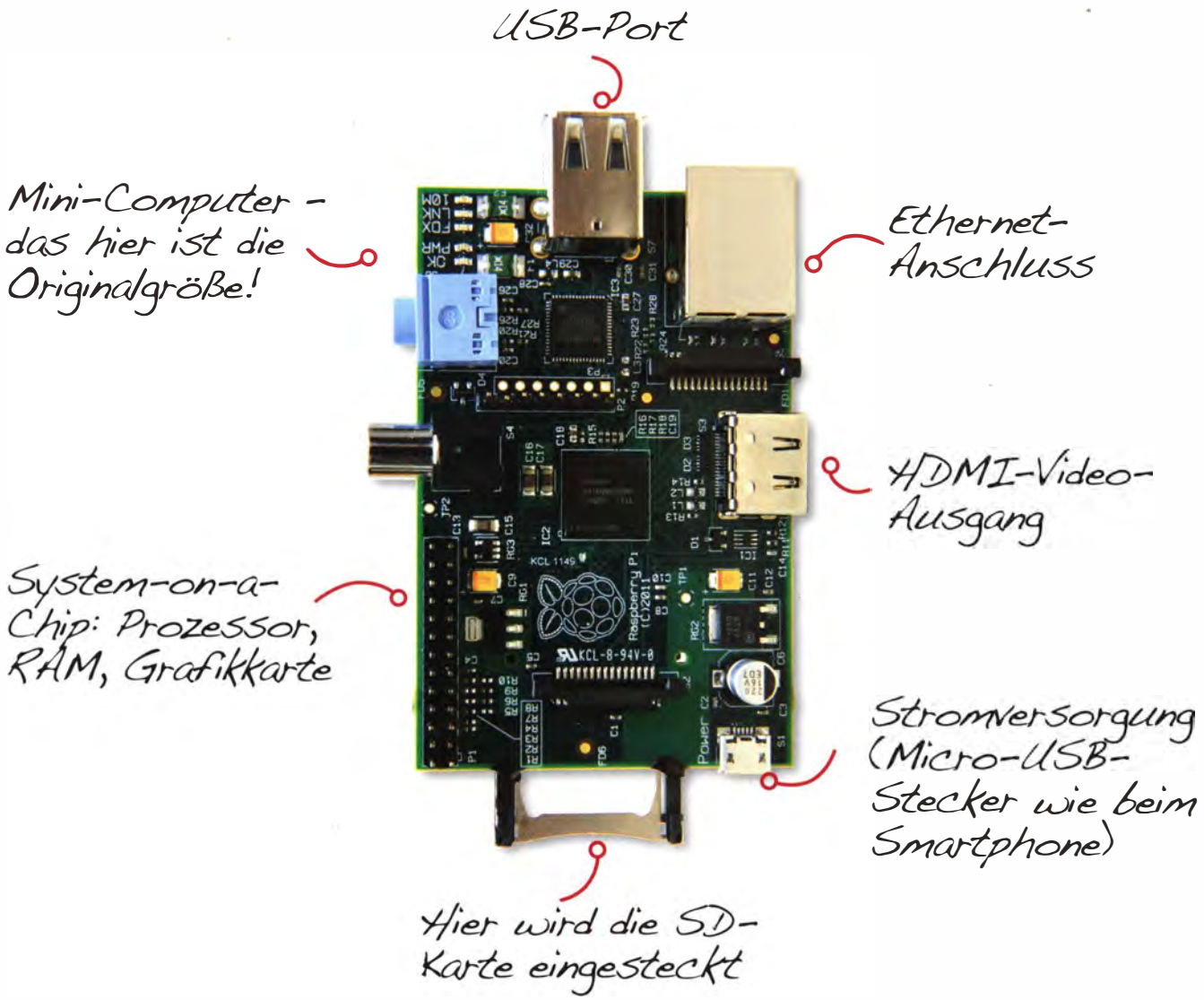
Falls Sie unsere Lektionen gelesen haben, um Ihren Raspberry Pi besser beherrschen und nutzen zu können, wird Ihnen das neue Wissen sehr weiterhelfen. Bleiben Sie am Ball! Probieren Sie Sachen aus! Falls Sie berufliche Pläne haben, für die gute oder exzellente Kenntnise von Linux erforderlich sind, und die entsprechenden Prüfungen ablegen möchten, informieren Sie sich über die Zertifizierungen des Linux Professional Institute. Die Informationen für Deutschland, Österreich, die Schweiz und andere europäische Länder finden Sie unter **www.lpice.eu/de/home.html**.

Tipps | Tricks | Anleitungen | Projekte | Spaß

FÜR EINSTEIGER UND PROFIS

Raspberry Pi:

Das Handbuch



Grundlagen
Installation, Anschlüsse und Zubehör einfach erklärt

Tutorials
Experimente und Projekte zum Nachmachen

Coding
Sofort loslegen mit Scratch, vertiefen mit Python

Linux
Einführung und achtteiliger Lehrgang vom Profi

Hardware
Der Pi als Retro-Konsole oder Mediencenter