grml - Linux fuer Systemadministratoren und Texttool-User

Michael Prokop
(Projektleiter des grml-Projektes,
mika@grml.org)

Revision: 24. Oktober 2005

Zusammenfassung

Dieses Dokument gibt im ersten Teil einen Einstieg in die Linux-Live-CD namens grml. Grundlegende Konzepte sowie Möglichkeiten von und mit grml werden erläutert. Im zweiten Teil werden die Technologien hinter grml beschrieben. Welche Möglichkeiten zur Hardwareerkennung existieren, kommen zum Einsatz und welche Technologien sind für Live-Systeme von Interesse.

Diese Version des Dokumentes bezieht sich auf grml 0.4 und grml-small 0.1 sowie die in Kürze erscheinenden Versionen grml 0.5, grml-small 0.2 und grml-usb 0.1.

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument entstand aus der Motivation, dass es zu den Themen grml, Linux als Live-System und Hardwareerkennung mit Linux kaum bzw. keine Dokumentation jenseits des Quellcodes gibt. Diese Arbeit soll diese Kluft schließen und dem Leser einen Einblick in grml geben.

2 Einführung

2.1 Was ist grml?

'Die Freiheit beschäftigt die Menschheit seit ihrer Entwicklung von der Rotte zum staatsbildenden Individuum. Beschäftigten sich in der Vergangenheit vorwiegend Philosophen, Staatstheoretiker und Politiker mit dem Thema, hat nun die IT-Branche dessen Relevanz für sich entdeckt.' [O'Reilly & Associates 2005]

So entstand im Herbst 2004 ein weiteres, neues OpenSource-Projekt namens "grml". grml ist eine kostenlos via Internet erhältliche Live-CD. 'Eine Live-CD ist eine bootfähige CD, welche ein Betriebssystem beinhaltet, das aber nicht installiert werden muss, um zu funktionieren.' [Wikipedia: Live-CD].

Die bekannteste auf Linux basierende Live-CD ist Knoppix von Klaus Knopper. Die Zielgruppe von Knoppix sind Ein- und Umsteiger, die sich die Möglichkeiten von "Linux ohne Risiko" anschauen möchten. Das Zielpublikum der hier behandelten Live-CD namens grml sind Fortgeschrittene und Experten. Sowohl für Systemadministratoren (die oft Systeme debuggen und reparieren müssen) als auch Textool-User (jene Personen, die gerne auf der Konsole arbeiten) sind die zur Arbeit notwendigen Applikationen vorhanden. Über 2200 Software-Pakete sind Teil von grml, über 800 davon sind welche, die man auf Knoppix nicht findet. In [Rankin 2005] ist zu lesen, dass 'Knoppix das Schweizer Taschenmesser der Live-CDs' ist. grml ist das Schweizer Taschenmesser der Systemadministratoren und Texttool-User.

2.2 Was braucht man für die Nutzung von grml?

Um grml zu nutzen empfiehlt sich die typische Computerausstattung in Form eines Computers mit CD-Laufwerk, Monitor sowie Eingabegeräte wie Tastatur und Maus. grml bringt eine automatische Hardware-Erkennung von Haus aus mit. Man muss sich zum Beispiel keine Sorgen darüber machen, welche Netzwerkkarte im Rechner steckt und welches Kernelmodul dafür notwendig ist. grml erkennt Hardware selbstständig und ohne manuelles Zutun. Natürlich kann durch manuelles Eingreifen - wie z.B. via einem speziellen Bootparameter - diese Erkennung gesteuert werden.

2.3 Was kann man mit grml machen?

grml eignet sich hervorragend für typische Systemadministratortätigkeiten. Darunter fallen Arbeiten wie Dateisystemchecks, Bootloader-Reparaturen, Hardware-Untersuchungen und Netzwerkdebugging. Aber auch für forensische Untersuchungen kommt grml zum Einsatz. Bei einer solchen Untersuchung wird 'ein forensisches Duplikat der betroffenen Datenträger erstellt'[Geschonneck 2004]. Dabei ist es wichtig, dass der betroffene Datenträger - im Normalfall eine Festplatte - absolut unverändert und unangetastet bleibt. Dies ist mit einem von CD laufenden grml-System möglich. Weitere dient grml dem Testen von Software sowie als eine risikoarme Test- und transportable Arbeitsumgebung. grml2hd ist ein Installationsprogramm, mit dessen Hilfe man ein laufendes grml-System auf die Festplatte schreiben und von dort nutzen kann. Innerhalb von wenigen Minuten und mit nur wenigen zu beantwortenden Fragen hat man auf diese Art und Weise ein benutzbares und mit Hardware-Erkennung ausgestattetes Linux-System zur Verfügung. grml kann also auch zum Prototyping und Testen von Hardware-Support seitens Linux herangezogen werden.

2.4 Das grml-Team

Die Projektleitung und Hauptentwicklung wird vom Autor dieses Dokumentes - Michael Prokop - ausgeführt. Weitere Hauptentwickler kümmern sich um zentrale Komponenten des grml-Systems. Der grml-Installer grml2hd zum Beispiel wird von Andreas Gredler betreut, Michael Gebetsroither kümmert sich unter anderem um den grml-Terminalserver. grml-Kontributoren wiederum sind Leute, die sich um spezielle Aufgabengebiete wie Applikationskonfiguration, Grafiken und Fotos sowie Accessibility/Zugänglichkeit aber auch Spezialgebiete wie udev kümmern.

2.5 Support

Kostenfreier Support ist über die englischsprachige [grml-Mailingliste] sowie den IRC-Kanal #grml auf irc.freenode.org zu beziehen. Speziell angepasste Versionen von grml, längerfristiger Support mit garantierter Reaktionszeit, Audits/Beratung, Vorträge, Workshops sowie Schulungen sind von den grml-Entwicklern gegen Entgeld zu beziehen. Mehr Details dazu gibt es auf solutions.grml.org.

2.6 News rund um grml

Ankündigungen zu neuen grml-Releases, neuen Teammitgliedern und anderen wichtigen Neuerungen rund um grml werden auf der [grml-Webseite]

bekanntgegeben. Zusätzlich zu der HTTP-Variante kann auch der [grml-RSS-Feed][1] abonniert werden.

Ein wichtiger Teil von grml ist die offene und transparente Entwicklung. Deswegen sind aktuelle News zur Entwicklung von grml im öffentlich verfügbaren [grml-Devel-Weblog] zu finden.

2.7 Informationen rund um grml

Der Hauptanlaufpunkt ist natürlich die [grml-Webseite], auf der es nicht nur allgemeine Informationen sowie eine [FAQ] (Frequently Asked Questions), sondern auch Informationen rund um die jeweiligen Releases gibt.

Seit dem 5. Juli 2005 gibt es zusätzlich auch das [grml-Wiki]. In diesem [Wiki] werden relativ "bewegliche" Informationen gesammelt. Angefangen von einer Todoliste, über aktuelle Informationen bis hin zu Tipps und Tricks ist dort alles Relevante zu finden. Das Wiki kann nicht nur von jedem gelesen, sondern auch bearbeitet werden. Mitarbeit ist nicht nur möglich sondern ausdrücklich erwünscht!

2.8 Die "grml-Geschmacksrichtungen"

Am 15. Mai 2005 hat Marc Haber in der Newsgroup de.comp.os.unix.linux.misc eine Debian-basierte Live-CD-Distribution mit einer ISO-Größe von maximal 55 MB gesucht [Posting von Marc Haber in dcoulm]. Es gab schon von grml-Entwicklern öfters den Ruf nach einer kleineren, spezialisierten grml-Version. Motiviert durch die Anforderungen von Marc Haber und durch Gespräche mit ihm wurde an einer "Miniversion" von grml gearbeitet. Am 5. Juli 2005 wurde dann grml-small in Version 0.1 veröffentlicht. grml-small ist ein rund 50MB großes ISO-Image und beinhaltet einen - im Vergleich zu grml(-large) - abgespeckten aber trotzdem aktuellen Kernel und ein paar grundlegende und für die Systemadministration wichtige Softwarepakete. Weder ein X-Server noch große Softwarepakete wie LaTeX oder Emacs sind an Board, die platzfressende Dokumentation wurde ausgespart und auch eine Festplatteninstallation via grml2hd ist (aktuell) nicht vorgesehen. Für Rescue-Zwecke ist es aber ausreichend und dank der geringen ISO-Größe passt grml-small sowohl auf Visitenkartenrohlinge als auch kleinere (<=64MB) USB-Sticks.

grml-usb ist ein weitere grml-Version, welche im Zeitraum September 2005 veröffentlicht wird. Während grml(-large) der Luxusdampfer ist, der alle Features bringt und grml-small ein minimalistisches System - vergleichbar mit einem Jetski - ist, stellt grml-usb die Mitte - als eine Art Motorboot - dar. grml-usb ist ein ISO-Image mit ca. 240 MB und ist - wie es der Name schon andeutet - für den Einsatz auf USB-Sticks angepasst. Ein 256MB-

[1] engl. Really Simple Syndication, eine XML-Datei um sich mit einem RSS-Reader auf dem Laufenden zu halten

USB-Stick reicht aus um sowohl das grml-ISO-Image als auch persönliche Dateien unterzubringen. Mit dem Skript [grml2usb] lässt sich ein grml-ISO -Image auch einfach und unkompliziert auf einen USB-Stick installieren. grml-usb bringt sowohl einen X-Server als auch viele Netzwerk- und Securitytools mit. Eine Festplatteninstallation via grml2hd ist möglich, nur große Softwarepakete wie LaTeX und Emacs mussten - aus offensichtlichen Gründen - eingespart werden.

Im Folgenden wird die 700MB Variante von grml als **das** grml bezeichnet, im Zweifelsfall wird es auch grml-large benannt. grml-small und grml-usb kennzeichnen sich natürlich durch die Bezeichnung nach dem Bindestrich.

2.9 Qualitätssicherung

Der Releasezyklus sieht öffentliche, stabile Releases alle zwei bis drei Monate vor. Abweichungen können sich durch die Einführung von speziellen Features ergeben, werden aber öffentlich angekündigt (siehe auch [grml-Roadmap]). Der Debian-Unstable-Zweig - auf dem grml basiert - befindet sich in ständiger Entwicklung. Auch die von grml unterstützte Hardware kann aufgrund ihrer Vielfältigkeit nicht vollständig und im Detail von den Entwicklern getestet werden. Aus diesem Grund wurde nach Erscheinen von grml Version 0.3 das Beta-Tester-Unterprojekt ins Leben gerufen. Teil dieses Projektes sind alle zwei bis drei Wochen erscheinende Entwicklungsversionen. Diese Schnappschüsse der aktuell stattfindenen Arbeit an der grml-CD werden von grml-Entwicklern, Kontributoren und Beta-Testern getestet und etwaige Bugs werden zentral gesammelt. Stabile Releases können dann freigegeben werden, sobald das 0-Bugs-Kriterium erfüllt wurde. Bugs können natürlich auch bei stabilen Releases vorkommen. Informationen, wie man einen Bug meldet, findet man im [grml-Wiki] unter http://wiki.grml.org/doku.php?id=bugs.

Diskussionen rund um neue Features und eventuelle Probleme finden auf der geschlossenen Development-Mailingliste statt. Alle für die Allgemeinheit relevanten Informationen werden dann in gekürzter und aufbereiteter Form im [grml-Devel-Weblog] veröffentlicht.

2.10 grml-Spezialitäten

2.10.1 Softwareauswahl

In den über 2200 Softwarepaketen steckt eine Vielzahl an Applikationen. Die Software stammt hierbei zu >90% aus dem Debianprojekt. Den Rest stellen Debian-Pakete aus Ubuntu, vereinzelten anderen Quellen und natürlich dem grml-Pool selbst dar. Die Software im grml-Pool wurde eigens für grml paketiert, der Einsatz ist an sich aber auch auf reinen Debian-

Systemen möglich. Software, die als interessant eingestuft wird, aber weder bei Debian, noch Ubuntu oder weiteren Quellen zu finden ist, wird von den grml-Entwicklern selbst paketiert und in grml integriert.

Natürlich gehören verschiedene Editoren zum Lieferumfang von grml. Angefangen bei **vim** und **nano** über **joe** und **emacs** bis hin zu dem Original-Vi-Port **ex-vi** ist alles dabei. Texttool-User bekommen mit dem Mailclient **mutt** und dessen Fork **mutt-ng**, dem Newsreader **slrn**, dem Instant-Messaging-Client **centericq** und dem IRC-Client **irssi** wertvolle Software zur Auswahl.

Natürlich sind auch Tools zur Datenrettung an Board. **gpart** errät Partitionstabellen, mit **recover** lassen sich Dateien auf ext2-Dateisystemen wiederherstellen. **salvage-ntfs** und **scrounge-ntfs** dienen zur Datenrettung auf Partitionen mit dem NTFS-Dateisystem. Mit **ntfsresize** lassen sich NTFS-Partition verkleinern und vergrößern.

Für Auditing gibt es ebenfalls eine Vielzahl an Software an Board. Mit **rats** lassen sich typische Programmierfehler in Programmen - die in C, C++, PHP, Perl oder Python geschrieben sind - identifizieren. Mit **pscan** kann man typische Securityfehler in C-Quellcode aufspürfen. **bass** ist ein so genannter "Bulk Auditing Security Scanner" mit dem man nach einer Vielzahl von Exploits suchen kann. **satan** ist eine Softwaresammlung um Netzwerke auf ihre Sicherheit hin zu überprüfen. Mit **smb-nat** lassen sich Audits für NetBIOS-Dateifreigben durchführen.

Nachdem grml auch für Netzwerkdebugging zum Einsatz kommt, ist eine Vielzahl an Netzwerkapplikationen zu finden. Obligatorische Software wie der Securityscanner **nessus** sowie **nmap** fehlen dabei natürlich nicht. Mit **hping** lassen sich spezielle Netzwerkpakete erzeugen. **doscan** wiederum eignet sich für das verteilte Scannen von großen Netzwerken.

Der Nachfolger von CVS namens **subversion** ist sowohl als Client als auch in Serverform vertreten. Der bekannte Webserver **Apache** ist in Version 1.3 und Version 2.0 zu finden. Wer Datenbanken braucht, bekommt mit **MySQL**, **PostgreSQL** und **sqlite** bekannte Vertreter der OpenSource-Datenbanken zur Verfügung gestellt. Clients für weitere Revisionssysteme wie **bazaar**, **darcs** und **tla** sind natürlich auch verfügbar.

2.10.2 zsh und zsh-lovers

Jene Shell die im interaktiven Bereich auf Linuxsystemen meistens zum Einsatz kommt ist die Bash. Während die Bash Grundfunktionalitäten durchaus bietet, gibt es für Poweruser eine noch interessantere Shell: die Z-Shell, kurz zsh[zsh-Webseite]. Die zsh ist sehr modular aufgebaut, bietet einen hochkonfigurierbaren Vervollständigungsmechanismus und eine mächtige Skriptingsprache. Die Aussage von Thorsten Zilm und Karsten Günther trifft den Nagel auf den Kopf: 'Ihre Anwendung ist aufgrund der Leistungsfähigkeit nicht unbedingt einfach; sie wird daher überwiegend von Spezialisten eingesetzt.'[Zilm, Günther 2004] Aus diesem Grund kommt die

zsh als die zentrale Shell bei grml zum Einsatz. Sie ist auf grml aber nicht nur die Standard-Loginshell, sondern sie wird auch als Interpreter für /bin/sh herangezogen. Da die zsh in ihrer Standardkonfiguration leider sehr minimal ist, wird sie auf grml mit einer erweiterten und angepassten Konfiguration ausgeliefert. So können praktische Funktionen wie der Vervollständigungsmechanismus sowie vordefinierte Shellaliases und -funktionen out-of-the-box genutzt werden. zsh-Anfänger können so die Mächtigkeit und Möglichkeiten der zsh in kurzer Zeit erlernen und nutzen. Details zu der Konfiguration gibt es auf der [grml-zsh-Webseite] und in der dort erhältlichen **grml-zsh-Referenzkarte**. Diese Referenzkarte dokumentiert vordefinierte Aliases, Funktionen und Einstellungen der zsh auf grml.

Die zsh bietet eine Vielzahl an eleganten Lösungen für den Alltag eines Systemadministrators und Texttool-Users. Viele dieser Beispiele werden im zsh-lovers-Projekt gesammelt, das Teil von grml ist. In der Manualpage **zsh-lovers** ('man zsh-lovers') sind viele Tipps und Tricks gesammelt und es soll dem interessierten Anwender eine Anlaufstelle für Probleme und ein Ideengeber sein. Mehr Informationen zu der zsh bei grml und dem zsh-lovers-Projekt gibt es auf der [grml-zsh-Webseite].

2.10.3 grml-etc

Das Paket grml-etc beinhaltet diverse Konfigurationsdateien, die vor allem eine verbesserte Standardkonfiguration mit sich bringen. So können Programme in einer für den User angenehmeren Konfiguration ausgeliefert werden.

2.10.4 grml-scripts

Im Paket grml-scripts werden verschiedene Skripte, für die ein jeweils eigenes Paket zu viel Aufwand bedeuten würde, in Form einer Skriptsammlung zusammengefasst.

2.10.5 grml2hd

Das grml-System kann auf Systemen ohne Festplatte gebootet werden. Allerdings kann man grml nicht nur von CD-ROM nutzen, sondern es gibt auch eine Möglichkeit um es auf eine Festplatte zu installieren. Das dafür zuständige Programm ist grml2hd. Dieses Programm kopiert das aktuell laufende System mit nur einigen wenigen Nachfragen auf die Festplatte. Auf diese Weise kann man ein Linuxsystem inklusive vollständiger Hardwareerkennung - diese lässt sich natürlich direkt via Live-CD testen - schnell und einfach aufsetzen. Ein Beispiel, um grml auf der Partition /dev/hda1 sowie den Masterbootrecord (MBR) in /dev/hda zu installieren:

```
# grml2hd /dev/hda1 -mbr /dev/hda
```

In weiterer Folge muss man nur mehr ein paar generelle Frage beantworten. Mehr ist nicht notwendig und wenige Minuten später ist das System schon bootfähig. Die Dauer der Installation hängt primär von der verwendeten Hardware - vor allem dem CD-ROM-Laufwerk - ab, mit einem schnellen CD-ROM-Laufwerk, ca. 256MB RAM und einer schnellen Festplatte lässt sich eine Installation innerhalb von 10 Minuten durchführen.

Weitere Informationen zu grml2hd gibt es auf [grml2hd-Webseite].

2.10.6 grml-terminalserver

Nicht nur via CD-ROM oder via Festplatte lässt sich grml nutzen. Man kann mit dem grml-terminalserver das grml-ISO auch über das Netzwerk booten. Einerseits indem man die PXE-Möglichkeit[Wikipedia: PXE] (engl. Preboot Execution Environment) des Rechners aktiviert oder indem man eine Diskette mit dem notwendigen Netzwerktreiber nutzt. Beide Varianten werden vom grml-terminalserver unterstützt. Mehr Details dazu auf [grml-terminalserver-Webseite].

2.10.7 Bootmöglichkeiten von grml

Der Vollständigkeit halber sollte noch erwähnt werden, welche Bootmedien von grml unterstützt werden. Wie bereits zu lesen war, kann man grml nicht nur von CD/DVD oder über das Netzwerk (via grml-terminalserver) booten. Auch das Booten von einem USB- oder Firewire-Gerät ist möglich. Das bedeutet, man kann grml z.B. via USB-Stick (siehe das Skript grml2usb), via USB-Festplatte oder einem DVD-Laufwerk, das via Firewire angeschlossen ist, booten. Selbstverständlich lässt sich grml auch auf einer Compact-Flashkarte nutzen.

2.10.8 Kernel

Mit Kernel 2.6 wurde das Entwicklungsmodell von den Kernelentwicklern abgeändert. Stabile Releases wie man es noch von Kernel 2.4 gekannt hat gibt es nicht mehr. Die Qualitätssicherung und einen ”brauchbaren” Kernel muss nun der Distributor zur Verfügung stellen. Obwohl mit den 2.6.x.y-Releases ein enormer Fortschritt zu bemerken ist, erfüllt man mit einem unveränderten Kernel von [kernel.org] - dem so genannten Vanilla-Kernel - nicht immer alle gewünschten Anforderungen. Aus diesem Grund gibt es auch bei grml einen angepassten Kernel.

Der Kernel umfasst einerseits aktuelle Bugfixes, aber auch zusätzliche Funktionalitäten wie Support für Microsoft PPP compression/encryption (MPPC/MPPE), das Reiser4-Dateisystem und Speakup-Support. Kernelmodule für viele Wireless-LAN-Netzwerkkarten wie zum Beispiel ipw2100/2200,

madwifi und ndiswrapper werden ebenfalls unterstützt. Weitere Details rund um den Kernel bei grml gibt es auf [grml-Kernel].

3 Die Technik hinter grml

3.1 Vorwort

Die Beschreibung der folgenden Technologien bezieht sich - sofern nicht explizit erwähnt - auf jene Implementierung, die in Debian GNU/Linux zum Zeitpunkt der Entstehung dieses Dokumentes zu finden ist.

3.2 Debian

Das Debian-Projekt ist eine Gemeinschaft von Individuen, die in Gemeinschaftsarbeit ein freies Betriebssystem namens Debian GNU/Linux - oder kurz einfach nur Debian - entwickeln. Der Name Debian GNU/Linux erklärt sich dadurch, dass das dem grml-System zugrunde liegende Betriebssystem Linux ist. Ein Großteil der sogenannten Coreutils (Software wie cp und tar) entstammt dabei dem GNU-Projekt (GNU's Not UNIX). Debian bietet eine große Auswahl an Software (über 15.000 Softwarepakete), öffentlich zugängliche Mailinglisten, Bug-Tracking-Systeme und frei verfügbare Installationsmedien. Weitere Informationen rund um Debian gibt es unter [Debian-Homepage: About Debian].

3.3 Paketmanagement

grml beinhaltet über 2200 Softwarepakete. Um den Überblick über diese große Anzahl an Paketen nicht zu verlieren und die über 123.000 Dateien und Verzeichnisse aktuell zu halten bedarf es eines Paketmanagements. Dieses Paketmanagement ermöglicht ein automatisierbares und einheitlich anzuwendendes Hinzufügen (Installation), Entfernen (Deinstallation) und Aktualisieren (Update/Upgrade) der gewünschten Software. Wichtig ist auch, dass Benutzer eine dokumentierte Schnittstelle und keinen WFM-Hack (Works For Me) vorfinden. Aus diesem Grunde wurde Debian als Basis gewählt. Das Paketmanagement von Debian (bekannt als DPKG/APT bzw. die Debian-Pakete mit der Dateiendung .deb) ermöglicht auch eine sehr gute Qualitätssicherung und -kontrolle. Details zu APT findet man in [APT-Howto].

3.4 Bootvorgang

Um die Hardwareerkennung zu verstehen, bedarf es auch grundlegenden Wissens über den Bootvorgang auf einem Live-CD-System. Beim Start von

grml ist als Erstes ein Bootsplash zu sehen. Dieser Bootsplash wird von isolinux geladen. isolinux ist ein Bootloader für Linux, der im 'No-Emulation-Modus' dem ISO 9660/El Torito-Standard folgt. Dadurch können das Limit des Floppy-Emulation-Modus und die Kompatibilitätsprobleme des Harddisk-Emulation-Modus umgangen werden.[isolinux-Homepage]

isolinux lädt dann den Linux-Kernel und dieser führt wiederum eine initiale Ramdisk - bekannt unter dem Namen initrd - aus. Um jedoch überhaupt auf das Root-Dateisystem zugreifen zu können, bedarf es je nach verwendeter Hardware verschiedene Treiber. Das Root-Dateisystem selbst ist mit SquashFS komprimiert, der dafür notwendige Treiber ist fix in den Kernel eingebaut. Wenn das grml-ISO aber zum Beispiel in einem SCSI-CD-ROM-Laufwerk liegt, muss man für das SCSI-Laufwerk den dafür passenden Treiber laden bevor man die komprimierte Datei überhaupt erst ansprechen kann. Die Ramdisk ermöglicht nun das Ausführen verschiedener Kommandos noch bevor das Root-Dateisystem angesprochen wird. Der Vorgang soll möglichst flexibel und mächtig sein, gleichzeitig soll die Ramdisk aber trotzdem klein sein, da sie ja in den Speicher geladen werden muss. Dies wird durch die Verwendung der statisch kompilierten Busybox-Shell erreicht. Auch die Einbindung von UnionFS findet im Bootstrapping in der initrd statt.

Zur Erstellung der initrd gibt es verschiedene Möglichkeiten. Der erste Schritt ist das Anlegen einer loop-back-Datei:

3.4.1 initrd

```
# dd if=/dev/zero of=/tmp/initrd bs=1024 count=5000
# mke2fs -b 1024 -N 8192 -F -q -m 0 /tmp/initrd
# mount -t ext2 -o loop /tmp/initrd /mnt/test
# cd /mnt/test ; rm -rf lost+found
```

In diesem Dateisystem werden dann die notwendigen Treiber, Skripte und Binaries platziert. Um Platz zu sparen wird die Datei noch mit gzip komprimiert. Der Kernel dekomprimiert die initrd dann und bindet sie als temporäres Root-Dateisystem ein. Dann wird die Datei /linuxrc ausgeführt und sobald diese terminiert, wird das temporäre Dateisystem wieder ausgebunden (engl. unmounted) und die eigentliche Hardwareerkennung beim Bootvorgang kann ausgeführt werden.

In Zukunft ist auch der Einsatz von initramfs-tools nicht auszuschließen.[initramfs] Bei Debian ist weiters ein neues init-System namens init-ng im Entstehen. grml verwendet für den Bootvorgang via grml-autoconfig zwar bereits spezielle Methoden wie Statusmeldungen und Parallelisierung, eine Integration von init-ng ist aber durchaus nicht ausgeschlossen und bereits angedacht.

3.5 Hardware-Erkennung

3.5.1 hotplug

Hotplugging nennt sich die Möglichkeit, Geräte zur Laufzeit dem System hinzu zu fügen oder zu entfernen. Während man eine solche Möglichkeit früher nur von PCMCIA kannte, wurde mit der Verbreitung von USB- und Firewire-Geräten sowie dem Ausblick auf PCI-Hotplugging der Wunsch, Hotplugging-Support im Kernel zu haben immer größer. Seit dem Kernel 2.4 ist Hotplugging ein Standard-Feature von GNU/Linux.

Damit nicht jeder Linux-Distributor selbst Shellskripte entwickeln muss, wurde im Rahmen des linux-hotplug-Projektes[hotplug-Webseite] unter Leitung von David Brownell eine Shellskript-Sammlung für das Hotplugging geschaffen. Bei der aktuellen Hotplugging-Implementierung namens hotplug werden neu erkannte Geräte dem Userland via dem Shellskript-Wrapper /sbin/hotplug zugänglich gemacht. Dieses Shellskript lädt auf einem Debian-System die zur Geräteklasse gehörenden Skripte in /etc/hotplug.d/. Diese Skripte wiederum kümmern sich darum, dass die passenden Treiber via modprobe geladen und via rmmod entladen werden.

Allerdings ist zu beachten, dass bei Verwendung von udev dessen udevsend noch zwischen Kernel und hotplug steht. Die manuelle Kontrolle mittels "cat /proc/sys/kernel/hotplug" zeigt, ob wirklich udevsend zum Einsatz kommt. Bei grml ist dies der Fall und udevsend leitet die Events an den udev-Daemon weiter.

Eine Zuweisung von einer Gerätenummer (engl. device-id) zu deren passenden Treiber (Kernelmodul) wäre in Shellskript-Form mit zu großem Overhead versehen. Daher wird jedem Kernelmodul entnommen, welche Geräte es unterstützt. Diese Information wird beim Lauf von depmod durch Suchen nach MODULE_DEVICE_TABLE() in den Treiberdateien in eine Tabelle geschrieben. Diese Tabellen sind dann unter /lib/modules/ $KERNELVERSION/modules.$(CLASS)map zu finden. Die hotplug-Implementierung kann dann anhand dieser Zuweisungen die Zuordnung vom Gerät zum passenden Treiber vornehmen. Aber nicht nur Treiber können ge- und entladen werden, auch Netzwerkinterfaces können automatisch hinauf- und hinuntergefahren und gleichzeitigt konfiguriert werden. Ebenso können externe Datenträger wie USB-Sticks mit Hilfe von Drittprogrammen (HAL und weitere) automatisch in das Dateisystem eingebunden und auf Wunsch ein Icon auf dem Desktop angelegt werden.

Kernel 2.6 stellt Hotplugging für alle auf sysfs-basierenden Subsysteme (Bussysteme und Treiber) zur Verfügung. Für jedes Gerät, das ein so genanntes kobject im sysfs registriert, wird ein Hotplug-Event ausgelöst. Obwohl der Name hotplug nur das Feature Hotplugging suggeriert, unterstützt hotplug auch das so genannte Coldplugging, also die Erkennung von Geräten, die nicht erst zur Laufzeit erkannt werden. Deswegen ist hot-

plug ein essentieller Teil der Hardwareerkennung bei grml.

Die Shellskript-Implementierung hat leider einen entscheidenden Nachteil: die modules.*map-Dateien werden zeilenweise mit Shellkomperatoren (engl. bash comparitors) geparst. Dies ist sehr zeitintensiv und besonders bei einer kurzen Bootzeit sehr störend. Diesem Problem wirken zwei weitere Implementierung entgegen: hotplug-perl und hotplug-ng.

Die Perl-Implementierung namens hotplug-perl nutzt für jedes Subsystem ein eigenes Perlmodul. Die Modulmappings (modules.*map) und die Blacklist werden nur einmal pro Instanz gelesen und als Array bzw. Hash zur weiteren Verarbeitung gespeichert.[hotplug-perl-Webseite]

hotplug-ng geht noch einen Schritt weiter und ist eine hotplug-Implementierung in der Programmiersprache C.[hotplug-ng-Webseite] Allerdings hat hotplug-ng noch keinen Einzug in Debian gefunden. Die Autoren von udev die gleichzeitig auch hotplug-ng entwickelt haben bezeichnen sogar selbst hotplug-ng für obsolet, da in Zukunft udev die relevanten Teile von hotplug vollständig übernehmen könnte.

Um den Bootvorgang zu optimieren, kommt bei grml-small 0.1 und auch bei den in Kürze erscheinenden Versionen grml 0.5 und grml-usb 0.1 hotplug-light zum Einsatz. Diese von Marco D'Itri entwickelte Variante beschleunigt den Bootprozess enorm, nutzt aber gleichzeitig die Features von hotplug kombiniert mit den Möglichkeiten von udev.

3.5.2 udev

udev ist ein dynamischer Userspace-Geräteverwalter. Es überwacht Hotplug-Events und anhand von Geräteinformationen aus dem sysfs (/sys) erstellt es dynamisch und zur Laufzeit Gerätedateien unter /dev. Der Vorteil der Auslagerung aus dem Kernel (wie es beim früheren Standard devfs der Fall war) ist eine hohe Flexibilität. Geräte können beliebig benannt werden und die Benennung ist konform der LSB (Linux Standard Base[2]).

Vor allem kann auch Persistenz in der Namensgebung erreicht werden, wenn Geräte de- und aktiviert werden. Wenn ein USB-Drucker angeschlossen wird, bekommt die Gerätedatei die entsprechend der [LANANA-Liste] vordefinierte Majornummer 180 und die erste freie Minornummer zugewiesen. Sofern noch kein anderer USB-Drucker den Platz belegt wird die erste Minornummer '0' vergeben. Was aber passiert wenn nun ein zweiter USB-Drucker angeschlossen wird? Dieser bekommt die Minornummer '1' zugewiesen. Wenn aber der nun als /dev/usb/lp1 anzusprechende zweite USB-Drucker unter /dev/usb/lp0 erwartet wurde und eine dementsprechende Zuordnung via /etc/fstab geplant ist, schlägt dies fehl. Durch die

[2] ein, von der Free Standards Group verabschiedeter Standard, mit dem Ziel, die verschiedenen Linux-Distributionen hinsichtlich Dateisystemstruktur und grundsätzlich notwendiger Bibliotheken zu vereinheitlichen, um eine zu starke Zersplitterung zu vermeiden.[Wikipedia: LSB]

Verwendung von udev, der Verwendung einer eindeutigen Geräte-Identität und einem definierten Mountpunkt (zum Beispiel /mnt/usbdrucker_raum42) kann sichergestellt werden, dass der gewünschte Drucker passend anzusprechen ist.

Zum udev-Paket gehören mehrere Applikationen. Der udev-Daemon udevd serialisiert die Hotplug-Events vom Kernel, so dass die Events im Userspace in der richtigen Reihenfolge ankommen. Seit Release 0.050 kann udev auch die Rolle des hotplug-Multiplexers selbst übernehmen. Dies bietet den Vorteil, dass hotplug-Aufruf in der passenden Reihenfolge für die Generierung der benötigten Gerätedateien geschieht.

udevsend leitet die die Events an den udev-Daemon. Das Tool scsi_id dient dazu, um die SCSI-Identifikation auszulesen und eine eindeutige Identifikationsnummer zu generieren. udev_volume_id wird im Normalfall von den udev-Regeln aufgerufen, um Gerätenamen passend zum Namen, UUID oder dem Dateisystemtyp einer Partition anzulegen.

Information zu einem Gerät können via udev mit dem Tool udevinfo abgefragt werden. Mit udevtest kann simuliert werden, was udev im Nicht-Testlauf ausführen würde. udevstart wiederum kommt - wie es schon der Name suggeriert - beim Start von udev zum Einsatz. udevstart geht durch das sysfs-Dateisystem und legt für jeden gefunden Eintrag ein Gerät im Geräteverzeichnis /dev an. So kann ein leeres /dev-Verzeichnis befüllt werden.

udev entwickelt sich sehr rasch und da ein gutes Zusammenspiel mit dem verwendeten Kernel vorhanden sein soll (teilweise ist sogar eine relativ aktuelle Kernelversion Voraussetzung zur vollständigen Funktion), nimmt das grml-Team die Paketierung von udev selbst vor. In Zukunft wird udev noch weitere Teile von hotplug übernehmen und es womöglich sogar obsolet machen.

Im Folgenden nun ein paar Beispiel des Einsatze von udev:

```
mika@grml ~ % vol_id -l /dev/hdc    % Dateisystemlabel auslesen
grml_0.4-1
mika@grml ~ % udevinfo -q path -n /dev/hdc  % sysfs-Pfad eines Gerätes anzeigen
/block/hdc
mika@grml ~ % udevinfo -q all -n /dev/hdc   % alle Werte zu einem Gerät anzeigen
P: /block/hdc
N: hdc
S:
S: cdrom
S: cdrw
S: dvd
mika@grml ~ % udevtest ‘udevinfo -q path -n /dev/hdc‘ % Testlauf durchführen
version 056
looking at ’/block/hdc’
opened class_dev->name=’hdc’
```

```
configured rule in '/etc/udev/rules.d/cd-aliases.rules[9]' applied, \
added symlink '%c{1} %c{2} %c{3} %c{4} %c{5} %c{6}'
creating device node '/dev/hdc', major = '22', minor = '0', \
mode = '0660', uid = '0', gid = '24'

mika@grml % udevinfo -a -p /sys/class/net/eth0   % detaillierte Informationen

udevinfo starts with the device the node belongs to and then walks up the
device chain, to print for every device found, all possibly useful attributes
in the udev key format.
Only attributes within one device section may be used together in one rule,
to match the device for which the node will be created.

 looking at class device '/sys/class/net/eth0':
 SYSFS{addr_len}="6"
 SYSFS{address}="11:22:33:44:55:66"
 SYSFS{broadcast}="ff:ff:ff:ff:ff:ff"
 SYSFS{features}="0x3"
 SYSFS{flags}="0x1002"
 SYSFS{ifindex}="4"
 SYSFS{iflink}="4"
 SYSFS{mtu}="1500"
 SYSFS{tx_queue_len}="1000"
 SYSFS{type}="1"

follow the class device's "device"
 looking at the device chain at '/sys/devices/pci0000:00/0000:00:09.0':
 BUS="pci"
 ID="0000:00:09.0"
 SYSFS{class}="0x020000"
 SYSFS{detach_state}="0"
 SYSFS{device}="0x9200"
 SYSFS{irq}="17"
 SYSFS{subsystem_device}="0x1000"
 SYSFS{subsystem_vendor}="0x10b7"
 SYSFS{vendor}="0x10b7"

 looking at the device chain at '/sys/devices/pci0000:00':
 BUS=""
 ID="pci0000:00"
 SYSFS{detach_state}="0"

grml@grml %
```

3.5.3 discover

Discover ist ein von der Firma Progeny Linux Systems Inc. entwickeltes Hardware-Identifikationssystem, das auf der Bibliothek libdiscover2 basiert. Discover besteht aus den drei Kommandozeilenapplikationen discover-

modprobe (Laden von Kernelmodulen), discover-config (Information über die Paketkonfiguration) und discover (die zentrale Hardwareerkennungs-Applikation). Die Implementierung für Linux wurde basierend auf dem Code von detect von MandrakeSoft SA durchgeführt.

Bei discover werden in XML-Dateien Informationen zu Geräten und ihrer Identifikation gespeichert. Es verwendet dabei für jeden Bustyp bis zu drei verschiedene Arten von Listen: eine Bustypliste (busclass-list), eine Herstellerliste (vendor-list) und eine Geräteliste (device-list). Discover unterstützt aktuell die Bustypen ATA, PCI, PCMCIA, SCSI und USB. Die Herstellerliste stellt nach außen hin Informationen zu einem bestimmten Gerät zur Verfügung. Gültige Gerätetypen wären zum Beispiel audio, bridge, display, network und printer. Die XML-Dateien sind auf einem Debian-System unter /lib/discover/ zu finden. In diesen wird nun zum Beispiel angegeben, welches Kernelmodul mit welchen Optionen für ein bestimmtes Gerät zu verwenden ist.

Ein Beispiel um Informationen über die verfügbaren Grafikkarten zu bekommen:

```
% discover -v --type-summary --disable-bus all --enable-bus pci display
Disabled ata
Disabled pci
Disabled pcmcia
Disabled scsi
Disabled usb
Enabled pci
Loading XML data... pci Done
ATI Technologies Inc RV350 AP [Radeon 9600]
ATI Technologies Inc RV350 AP [Radeon 9600] (Secondary)
```

Weite Details zu discover sind in der offiziellen Dokumentation auf der [Discover-Homepage] zu finden.

3.5.4 hwinfo

hwinfo ist ein auf der Hardware-Detection-Bibliothek (libhd) aufbauendes Hardwareerkennungstool der Firma SuSE. hwinfo steht unter der GPL und kommt bei grml aktuell nur bei der Grafikkartenerkennung bei grml-x zum Einsatz. Eine weitergehende Integration in den Prozess der Hardwareerkennung ist aber nicht auszuschließen. hwinfo liest seine Hardwareinformationen nämlich anders als zum Beispiel kudzu (das bei grml aber ebenfalls nicht mehr eingesetzt wird) nicht aus einer externen Datei aus, sondern bringt es in der libhd mit. Dies hat zwar den Nachteil, dass eine Aktualisierung der Hardwareinformationen (diese wird in den IDS-Dateien vorgenommen) ein Neubauen der Software bedingt. Es bringt umgekehrt aber einen Performancevorteil, der gerade beim Bootvorgang nicht zu vernachlässigen ist.

Wie funktioniert hwinfo nun im Groben? Anhand eines Beispieles soll dies nun nachvollzogen werden. Wir interessieren uns für die Grafikkarte und möchten Herstellerinformationen über diese, aber auch den Namen des zu ladenden Treibers bekommen. lspci liefert uns Informationen über Geräte die an den PCI-Bus angeschlossen sind:

```
% lspci | grep VGA
0000:01:00.0 VGA compatible controller: ATI Technologies Inc Rage Mobility P/M A
% lspci -n | grep "0000:01:00.0"
0000:01:00.0 0300: 1002:4c4d (rev 64)
```

Nachdem wir nun die PCI-ID kennen, vergleichen wir den Output einmal mit jenem von hwinfo:

```
% hwinfo --gfxcard | grep -A3 " Vendor:"
 Vendor: pci 0x1002 "ATI Technologies Inc"
 Device: pci 0x4c4d "3D Rage P/M Mobility AGP 2x"
 SubVendor: pci 0x104d "Sony Corporation"
 SubDevice: pci 0x80f6
%
```

Nun der dazu passende Ausschnitt aus den IDS-Dateien des hwinfo-Quellcodes:

```
 vendor.id pci 0x1002
&device.id pci 0x4c4d
+device.name 3D Rage P/M Mobility AGP 2x
+driver.xfree 4|ati
```

Die Information darüber wird wirklich aus der libhd gelesen, wie uns das nachfolgende Kommando beweist:

```
% strings /usr/lib/libhd.so | grep "3D Rage P/M Mobility AGP 2x"
3D Rage P/M Mobility AGP 2x
%
```

3.5.5 D-BUS

D-BUS ist ein Nachrichten-Bussystem.[D-BUS] Ein Systemdaemon kümmert sich um Systemnachrichten wie 'neue Hardware wurde hinzugefügt'. Ein Userspace-Dämon (per-user-login-session daemon) kümmert sich um die Interprozesskommunikation (engl. IPC - Inter Process Communication) zwischen den Userlandapplikationen. D-BUS bietet optional Bindings für GLib, Qt, Python und Mono/.NET. Projekte die auf D-BUS aufsetzen, sind unter anderem udev und HAL.

3.5.6 HAL

Die Idee hinter [HAL] (Hardware Abstraction Layer, engl. Hardware Abstraktionsschicht) ist es, die Hardware-Ebene zu abstrahieren und in Form einer Bibliothek für Software zur Verfügung zu stellen. So muss sich Software nicht um Details rund um die Hardware und deren Verwendung kümmern. Applikationen können dadurch plattformübergreifend geschrieben und verwendet werden. Sobald der HAL-Daemon (hald) und das D-BUS-System laufen, kann man sich mit 'lshal' Hardwareinformationen anzeigen lassen. Ein Beispiel der Anwendung um Prozessor-Informationen zu beziehen:

```
% lshal 2>/dev/null | sed -ne "/^udi = .*Processor/,/^$/p"
lshal version 0.4.7
udi = '/org/freedesktop/Hal/devices/Processor'
 linux.procfs.cpuinfo.bogomips = '3578.26' (string)
 linux.procfs.cpuinfo.flags = 'fpu vme de pse tsc msr pae mce [...]
 linux.procfs.cpuinfo.wp = 'yes' (string)
 linux.procfs.cpuinfo.cpuid_level = '1' (string)
 linux.procfs.cpuinfo.fpu_exception = 'yes' (string)
 linux.procfs.cpuinfo.fpu = 'yes' (string)
 linux.procfs.cpuinfo.coma_bug = 'no' (string)
 linux.procfs.cpuinfo.f00f_bug = 'no' (string)
 linux.procfs.cpuinfo.hlt_bug = 'no' (string)
 linux.procfs.cpuinfo.fdiv_bug = 'no' (string)
 linux.procfs.cpuinfo.cache_size = '256 KB' (string)
 linux.procfs.cpuinfo.cpu_mhz = '1795.462' (string)
 linux.procfs.cpuinfo.stepping = '1' (string)
 linux.procfs.cpuinfo.model_name = 'AMD Athlon(tm) XP 2200+' (string)
 linux.procfs.cpuinfo.model = '8' (string)
 linux.procfs.cpuinfo.cpu_family = '6' (string)
 linux.procfs.cpuinfo.vendor_id = 'AuthenticAMD' (string)
 linux.procfs.cpuinfo.processor = '0' (string)
 info.parent = '/org/freedesktop/Hal/devices/computer' (string)
 info.product = 'Processor' (string)
 info.bus = 'unknown' (string)
 info.capabilities = 'linux.sysinfo' (string)
```

Via 'lshal –monitor' kann man die Geräteliste überwachen. Nützlich ist HAL in Kombination mit hardwarenahen Applikationen wie udev und hotplug.

3.6 squashfs

SquashFS ist ein komprimiertes read-only-Dateisystem für Linux. SquashFS komprimiert Daten, Inodes und Verzeichnisse und kann dabei Dateien sowie Dateisysteme bis zu einer Größe von 4 GB verarbeiten. Dabei werden

Blockgrößen von bis zu 64 KB verwendet, dies ermöglicht bessere Komprimierungsraten im Vergleich zu normalen Blockgrößen wie 4KB. Dateiduplikate werden dabei von SquashFS erkannt und entfernt.

Für SquashFS-Support benötigt man den SquashFS-Linuxkernel-Patch und das Tool mksquashfs. Als ersten Schritt lädt man das SquashFS-Modul via insmod bzw. modprobe. Dann überprüft man am Besten, ob der Kernel SquashFS unterstützt:

```
% grep squashfs /proc/filesystems
        squashfs
```

Nun kann man via folgendem Kommando aus dem Verzeichnis /ein/-verzeichnis die SquashFS-Datei test.sqsh erzeugen:

```
% mksquashfs /ein/verzeichnis test.sqsh
```

Die Datei test.sqsh ist vom Speicherbedarf her nun wesentlich kleiner als das Verzeichnis /ein/verzeichnis. Diese Technik macht es möglich, dass bei grml rund 2.1 GB auf einen 700 MB-Rohling passen. Um diese SquashFS-Datei nun zu nutzen bindet man sie in das Dateisystem wie folgt ein:

```
% mount test.sqsh /mnt/test -t squashfs -o loop
```

Weitere Informationen zu SquashFS findet man auf der [squashfs-Webseite] sowie im [SquashFS-HowTo].

3.7 unionfs

'Typischerweise besteht ein komplettes Linux-System aus mehreren Tausend bis hin zu einigen Millionen Dateien, die zum Speichern von Programmen, Daten und allen Arten von Informationen verwendet werden. Bekanntlich werden hierarchische Verzeichnisstrukturen verwendet, um Dateien zu katalogisieren und zu gruppieren. Es gibt verschiedene Ansätze, um die dazu notwendigen Strukturen zusammen mit den Daten permament zu speichern.'[Mauerer 2004] Gängig sind hierbei Dateisysteme wie ext2/ext3, reiserfs und XFS. Klassische Dateisysteme unter Linux werden als Ganzes in eine bestehende Verzeichnisstruktur eingebunden. Die ursprünglichen Inhalte eines Verzeichnisses werden dabei ausgeblendet. Die Kombination von zwei Dateisystemen ist also nicht einfach möglich. Um dies trotzdem zu ermöglichen, gibt es so genannte transluzente Dateisysteme.

'Ein transluzentes Dateisystem baut auf zwei anderen Dateisystemen auf und vereinigt deren Inhalte in besonderer Form: Auf das erste der beiden unterliegenden Dateisysteme wird nur lesend zugegriffen und es dient als Basis für das transluzente Dateisystem. Das zweite Dateisystem erlaubt

zusätzlich Schreibzugriffe und hat die Aufgabe neue und veränderte Dateien aufzunehmen. Ansonsten verhält sich das Letztere für den Benutzer durchsichtig daher der Name transluzent (translucent, engl: durchsichtig).' [Zitterell 2005]

Durch diese Technologie ist es möglich, ein read-only-Medium wie zum Beispiel eine CD-ROM mit einer RAM-Disk (Bereich in einem temporären Speicher [RAM oder Swap]) zu vereinen und virtuell schreibbar zu machen. Unionfs ist ein solches transluzentes Dateisystem und ermöglicht es, eine RAM-Disk tranparent über das Dateisystem zu legen und das System somit wie eine Festplatteninstallation nutzbar zu machen. Der Name rührt daher, dass Unionfs mehrere Verzeichnisse in einer 'unified view' (einzelnen Sicht) vereint.

Transluzente Dateisysteme können dabei in zwei verschiedenen Arten implementiert werden: durch Manipulation von Systemaufrufen oder durch die Verwendung des virtuellen Dateisystemlayers VFS des Kernels. Die Variante mit Systemaufrufen hat aber folgende Nachteile:

- Dateioperationen des Kernels werden nicht berücksichtigt
- seit Kernelversion 2.6 ist eine Manupulation der Systemaufrufe nicht mehr ohne Änderung der Kernelquellen möglich
- großer Overhead durch Überprüfen der den Systemcalls übergebenen Zeichenketten (Probleme zum Beispiel mit Symlinks)
- das Behandeln konkurrierender Prozesse ist problematisch
- Caching kaum bzw. sehr schwer möglich

Aus diesem Grund setzt Unionfs auf dem VFS auf. Unionfs ist dabei nicht die einzige Implementierung eines transluzenten Dateisystems. FreeBSD bietet schon seit längerem ein sogenanntes 'union filesystem', dieses ist aber relativ schlecht dokumentiert und hat mit dem hier behandelten UnionFS nichts zu tun. Für SUN Solaris gibt es mit tfs auch ein äquivalentes Projekt. Weitere transluzente Dateisysteme sind Translucency von Bernhard Wiedemann, mini_fo sowie cowfs als Teil der shadowfs-LD_PRELOAD-Wrapper-Sammlung von Clifford Wolf. UnionFS ist relativ jung, aber aktuell die am weitesten fortgeschrittene Implementierung eines transluzenten Dateisystems.

Um Unionfs zu nutzen benötigt man das Unionfs-Kernelmodul. Via 'modprobe unionfs' lädt man das Modul. Ob das Modul wirklich geladen ist kann man via 'lsmod | grep unionfs' überprüfen. Bei grml wird Unionfs bereits in der initrd-Phase geladen, da direkt nach dem Zugriff auf die SquashFS-Datei Unionfs transparent über das Root-Dateisystem ('/') gelegt wird.

Ein Beispiel, wie man aus einem readonly-Verzeichnis namens /GRML mit dem Darüberlegen eines schreibbaren Verzeichnisses (/ramdisk) mittels Unionfs ein schreibbares Verzeichnis erreichen kann:

```
% mount -t unionfs -o dirs=/ramdisk=rw:/GRML=ro none /UNIONFS
```

Ein weiteres Beispiel demonstriert, wie man die Verzeichnisse /dvd1 und /dvd2 gemeinsam in einem dritten Verzeichnis /mnt/allinone vereinen kann:

```
mount -t unionfs -o dirs=/dvd1:/dvd2 none /mnt/allinone
```

Unionfs steht aktuell noch in aktiver und ständiger Entwicklung. Mit weiteren Funktionen und einer erhöhten Stabilität kann in Zukunft also gerechnet werden.

4 Zusammenfassung

Wir haben in diesem Dokument gezeigt, dass grml eine mächtige Live-CD ist, die sowohl für Systemadministratoren als auch für Texttool-User sehr gut geeignet ist. Der Bereich der Hardwareerkennung ist ein sich ständig weiterentwickelndes Betätigungsfeld. Neuartige Technologien müssen sorgfältig überprüft, getestet und in bestehende Systeme eingefügt werden. Nur dann ist ein stabiler und zufriedenstellender Betrieb möglich.

Feedback, Ergänzungen und Korrekturen zu diesem Dokument werden vom Autor gerne entgegen genommen! grml.org/contact/

Literatur

[Rankin 2005] Kyle Rankin: "KNOPPIX HACKS"; O'Reilly Media Inc., Gravenstein Highway North (2005)

[Mauerer 2004] Wolfgang Mauerer: "Linux Kernelarchitektur"; Carl Hanser Verlag, München/Wien (2004)

[Peikari, Chuvakin 2004] Cyrus Peikari und Anton Chuvakin: "Kenne deinen Feind"; O'Reilly Verlag GmbH & Co. KG, Köln (2004)

[O'Reilly & Associates 2005] O'Reilly & Associates: "Open Source - kurz & gut"; O'Reilly Verlag GmbH & Co. KG, Köln (1999)

[Zilm, Günther 2004] Thorsten Zilm und Karsten Günther: "Bash gepackt"; mitp-Verlag, Bonn (2004)

[Geschonneck 2004] Alexander Geschonneck: "Computer Forensik"; dpunkt.verlag GmbH, Heidelberg (2004)

[APT-Howto] `http://www.debian.org/doc/manuals/apt-howto/index.de.html` (25.08.2005, Debian)

[Debian-Homepage: About Debian] `http://www.debian.org/intro/about` (01.06.2005, Debian)

[D-BUS] `http://hal.freedesktop.org/wiki/Software_2fdbus` (26.04.2005, freedesktop.org)

[Discover-Homepage] `http://componentizedlinux.org/discover/documentation/guide.html` (25.08.2005, Progeny)

[FAQ] `http://grml.org/faq/` (25.08.2005, grml)

[grml2hd-Webseite] `http://grml.org/grml2hd/` (25.08.2005, grml)

[grml2usb] `http://grml.org/files/scripts/grml2usb` (25.08.2005, grml)

[grml-Devel-Weblog] `http://grml.supersized.org/` (25.08.2005, grml)

[grml-Kernel] `http://grml.org/kernel/` (25.08.2005, grml)

[grml-Mailingliste] `http://grml.org/mailinglist/` (25.08.2005, grml)

[grml-Webseite] `http://grml.org/` (25.08.2005, grml)

[grml-Wiki] `http://wiki.grml.org/` (25.08.2005, grml)

[grml-Roadmap] http://grml.org/roadmap/ (25.08.2005, grml)

[grml-RSS-Feed] http://grml.org/index.rss (25.08.2005, grml)

[grml-zsh-Webseite] http:/grml.org/zsh/ (25.08.2005, grml)

[grml-terminalserver-Webseite] http:/grml.org/terminalserver/ (25.08.2005, grml)

[Posting von Marc Haber in dcoulm] http://groups.google.at/group/de.comp.os.unix.linux.misc/msg/5220e8d133791129 (15.05.2005)

[HAL] http://hal.freedesktop.org/wiki/Software_2fhal (25.08.2005, freedesktop.org)

[hotplug-Webseite] http://linux-hotplug.sourceforge.net/ (25.08.2005, linux-hotplug)

[hotplug-perl-Webseite] http://opensource.idealcorp.com/hotplug-perl/ (25.08.2005, I.D.E.A.L. Technology Corporation)

[hotplug-ng-Webseite] http://kernel.org/pub/linux/utils/kernel/hotplug/ (25.08.2005, kernel.org)

[initramfs] http://lwn.net/Articles/14776/ (09.06.2005, LWN.net)

[isolinux-Homepage] http://syslinux.zytor.com/iso.php (25.08.2005, H. Peter Anvin)

[kernel.org] http://kernel.org/ (25.08.2005, kernel.org)

[LANANA-Liste] http://www.lanana.org/docs/device-list/devices.txt (25.08.2005, The Linux Assigned Names And Numbers Authority)

[SquashFS-HowTo] http://www.artemio.net/projects/linuxdoc/squashfs/SquashFS-HOWTO.html (25.03.2005, Artemiy I. Pavlov)

[squashfs-Webseite] http://squashfs.sourceforge.net/ (25.08.2005, squashfs.sourceforge.net)

[Wikipedia: Live-CD] http://de.wikipedia.org/wiki/Live-CD (22.05.2005, Wikipedia)

[Wikipedia: PXE] http://de.wikipedia.org/wiki/Preboot_Execution_Environment (25.08.2005, Wikipedia)

[Wikipedia: LSB] http://de.wikipedia.org/wiki/Linux_Standard_Base (25.08.2005, Wikipedia)

[Wiki] http://de.wikipedia.org/wiki/Wiki (28.08.2005, Wikipedia)

[Zitterell 2005] http://www.ks.uni-freiburg.de/download/studienarbeit/WS04/01-05-Studienarbeit-ThZitterell.pdf (Jänner 2005)

[zsh-Webseite] http:/www.zsh.org/ (25.08.2005)

[zsh-Userguide] http://zsh.sunsite.dk/Guide/zshguide.html (02.06.1999)

3 19 3 7 4 10 10 17 16 5 9 6 5, 6 10 4 4, 5 5, 6 6 5 8 9 5 18 12 13 13 11 11 9 13 19 19 3 9 13 5 20 7

5 Revisisionen dieses Dokumentes

Mon Oct 24 13:32:59 CEST 2005: Rechtschreibfehler nach Hinweis durch Gert Below korrigiert
Thu Sep 15 21:50:49 CEST 2005: Hinweis auf Drittprogramme im hotplug-Teil, Verbesserung der Begriffserklärung sowie Anpassung von udev_volume_id auf vol_id im udev-Teil nach Hinweis durch Tobias Klauser
Wed Sep 14 22:55:29 CEST 2005: einige Stilverbesserungen, Rechtschreibkorrekturen nach Hinweis durch Nico Golde
Tue Sep 13 12:06:57 CEST 2005: Rechtschreibfehler korrigiert, Textvorhebungen und Verweis auf grml-zsh-Referenzkarte nach Hinweis durch Marcel Cedric
Mon Sep 12 01:16:48 CEST 2005: Zwei Rechtschreibfehler nach Hinweis durch Ronny Plattner beseitigt
Thu Sep 01 11:12:41 CEST 2005: Veröffentlichung der 1. Revision