MAC OS X

unter der Haube

Vorwort

Bei diesem kleinen (aber feinen) Büchlein handelt es sich um die Abschlußarbeit von Claus-Peter Frenzel und Daniel Windelband in dem Modul Netzwerktechnik der Weiterbildung zum staatlich geprüften Techniker für Medien an der BBS Neustadt an der Weinstraße.

Es soll die Gemeinsamkeiten, Vor- und Nachteile der OS MacOS X und Linux aufzeigen und versuchen näher zu bringen. Wobei dem Kenner hier bereits die Erste Unstimmigkeit auffallen wird:
Während es sich bei OS X um ein (wie der Name schon sagt) Operating system also einem Betriebssystem mit grafischer Oberfläche (GUI - Graphical-User-Interface) handelt ,kann man bei Linux eigentlich nur vom Kernel, also dem Herzen eines OS, sprechen. Doch dazu mehr in Inneren des Buches.

Da es sich um eine sehr tiefgreifende und spezielle Thematik handelt gehen wir davon aus dass sich kein „Greenhorn" damit beschäftigt und deshalb auch nicht mehr auf sogenannte „Basics" eingehen werden. Dies hätte sicher den Rahmen der uns zur Verfügung gestellten Zeit gesprengt.

Alle hier beschriebenen Befehle wurden von uns getestet, wobei wir allerdings festgestellt haben, dass es, abhängig vom Systemupdate und des verwendeten Prozessors, Unterschiede bestehen und deshalb eventuell einige Befehle nicht an jedem Rechner funktionieren. Als Grundlage diente und die Systemversion 10.3.8 mit Darwin 7.8 beim MacOSX und System V bei Linux.
Für die Richtigkeit der Befehle, deren Anwendung und eventuellen Reaktionen (unter Umständen auch Schäden des Systems) können wir leider keine Haftung übernehmen.

In diesem Sinne wünschen wir allen die sich dieser Thematik widmen viel Spaß und Kurzweil beim Lesen dieser Ausarbeitung.

Die Autoren

Claus-Peter Frenzel & Daniel Windelband

Non quia difficilia sunt non audemus,
sed quia non audemus difficilia sunt.
Seneca

(Nicht weil es schwer ist, wagen wir es nicht,
sondern weil wir es nicht wagen, ist es schwer)

Inhaltsverzeichnis

Linux

Einleitung **7**

Geschichtliches **7**

Die Verzeichnisstruktur **9**

Der Bootvorgang **12**

Von einer Diskette booten 12
Lilo benutzen 14
Lilo als sekundärer Bootloader 18
Die Bootoptionen 18
Lilo entfernen 20

Systemstart **21**

Boot-Meldungen des Kernels 21
init und inittab 22
Runlevel 23
Definition der Runlevel 23

Zugriffsrechte **27**

User 27
Group 27
Others 27
Die Arten der Rechte 27
Zugriffsrechte ändern 28

BSD 30

Die BSD-Lizenz 30
Neuerungen von Berkeley 30
Die Anfänge 31
VAX Computer und DARPA Einfluss 31
Aufstieg 32
Fall 32
Weiterentwicklungen 32
BSD und AT&Ts Unix-Quellcode 32
Die BSD Projekte 33
Gemeinsamkeiten und Unterschiede 34

Mac OS X

Einleitung **36**

Architektur **36**

Die Systemprogramme 36
Die Benutzeroberfläche 36
Die Programm-Frameworks 36
Grafiken und Medien 37
Systemverwaltung 37
OS Grundlagen 37

Geschichtliches **37**

Copland 37
NEXTSTEP 38
Rhapsody 38
Classic 38
Java 38

Endlich ein neues Betriebsystem **39**

Grundlagen

Mach 3.0 Microkernel 40
Shell-Script 40
swap-Partition 40

Bootvorgang **40**

BootROM 40
POST 40
Open Formware 41
BootX 41
Systeminitialisierung 42

rc.boot 43
rc 44
System Starter 44
Login Window 45
System-Profiler 45
Das Filesystem 46
Type- und Creator-Codes 46
Bundles 47
Journaling File System 47
Fragemntierung 47
Weitere Filesysteme 48
Netzwerkbasierende Filesysteme 48
Verzeichnisstruktur 49
Zugriffsrechte 52
Zugriffsrechte vergeben 53
Das Sticky Bit 54
Benutzer 54
Gruppen 54
Benutzertypen 55
Benutzer verwalten 56
Benutzer anlegen 56
Netinfo-Manager 56
Nicht-menschliche Benutzer 56
Die Systemaktivität 57
Protokolldateien 57
Arbeiten mit Protokolldateien 57
Prozesse 58
Aktivität der Prozesse 58
Prozesse beenden 59
Open Directory 59
Verzeichnisdienste 60
Informationen der Open Directory 60
Verzeichnis-Domains 61
Die Shadow-Datenbank 61
Drucken 63
Cups 63
Druckmanager und Cups 64
Drucken von der Kommandozeile 64
Apple Talk Printer 65
Post-Script 65
PDF 65

Netzwerkverbindungen 66
Allgemein 66
Remote Login 66
ssh 67
ftp 67
Alltagstipps 68
Das Schriftenproblem 68
Nicht-startende programme 68
Rechtestruktur wiederherstellen 68
Festplatte reparieren 68
Löschen des NVRams 68
Open Firmware zurücksetzen 69
Unix-Wartungsarbeiten 69
Rücksetzen des Displays bei „books“ 69
Passwort vergessen 69
Quellenanganben 70

LINUX

Das Betriebssystem Linux ist derzeit in aller Munde. Es gibt wohl keine Fachzeitschrift, die diesem Betriebssystem nicht einen umfangreichen Artikel oder sogar ein komplettes Sondermagazin gewidmet hat, außer vielleicht den Mac Zeitschriften.

Linux ist ein 32 bit **Multi-User-Operating System** für **Intel-PC's**, oder auch für **Alpha Systeme**, die im Gegensatz zu Microsofts Windows echte 32-bit sind. Doch dazu später mehr.
Linux ist im Gegensatz zu den meisten anderen Betriebssystemen kein kommerzielles Produkt, sondern ein von auf der ganzen Welt verstreuten Entwicklern geschaffenes Betriebssystem. Diese Gruppe von Menschen hat die Ressourcen des Internets für ihr Projekt genutzt um Linux zur Verfügung zu stellen und entwickeln dieses Betriebssystem tagtäglich auf freiwilliger Basis weiter.

Das Fundament legte der Finne Linus Torvalds, der das Betriebssystem in seiner Universität entwickelte. Linux sollte **Minix** ersetzen und um ein großes Stück leistungsfähiger sein. Minix und Linux sind Unix-Clones, die durch Sicherheit und Stabilität zum Mammut in der IT-Branche geworden sind und deshalb weitläufig bekannt sein sollten. Linus veröffentlichte sein Werk und rief eine Initiative ins Leben: Linux und dazugehörige Applikationen stehen unter der GNU-License. Die GNU-License sieht vor, dass das Programm völlig kostenlos ist und der Quell-Code veröffentlicht und allen zugänglich gemacht werden muss - also ein Open-Source-System. Diese Tatsache bringt uns wieder zur Anfangs erwähnten weltweiten Weiterentwicklung. Gerade der GNU-License hat es Linux zu verdanken, dass es sehr stabil, modern und benutzerfreundlich ist. Auch die Hardwareeinbindung geht somit immer schneller von Statten. So kann sich jedermann mir den entsprechenden Kenntnissen über Linux und C, den Quellcode besorgen und ihn nach seinen Gedünken verändern und wieder anderen zur Verfügung stellen.

Hier eine kleine Linux History:

- Ken Tomphson entwickelte 1969 UNIX. Der Name kommt aus Uni von Universität und X als Homophon zum MULTICS-Betriebssystem zustande.
- BSD UNIX wurde 1978 entwickelt. Hauptsäclich anwenderfreundlicher gestaltet.
- Linux, vom damals 23 jährigen Linus Torvalds, 1991 ins Web gestellter UNIX-Clone.

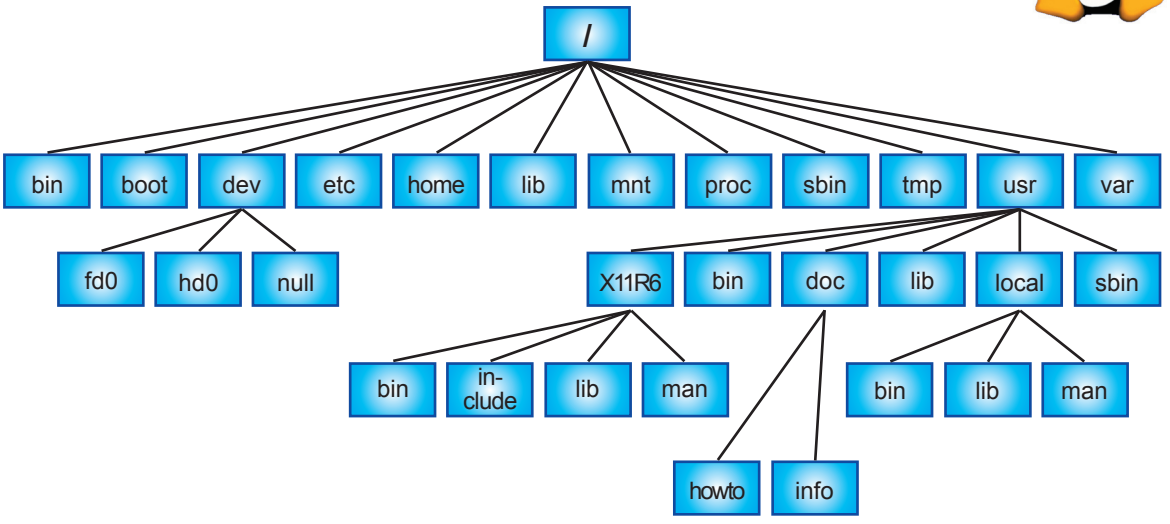
Streng genommen ist Linux selbst kein Betriebssystem, sondern der Kernel, die wichtigste Komponente des Betriebssystems. Dieser Kernel wird als Grundlage für Betriebssysteme verschiedenster Art verwendet.
Im Sprachgebrauch der Free Software Foundation wird die bekannteste Betriebssystemvariante, die aus dem Linux-Kernel und Betriebssystemkomponenten von GNU kombiniert wird als GNU/Linux bezeichnet. Das GNU Projekt ermöglichte eigentlich erst den Linux Kernel. GNU stellt beispielsweise einen Compiler und andere UNIX typische Tools zur Verfügung.

Linux erhält man vorwiegend als Distribution. Eine Distribution besteht meist aus einem

speziell angepassten Linux-System und ist eine Zusammenstellung hauptsächlich freier Software zum Zwecke der Weitergabe. Sie enthält den Linux Kernel, die GNU-Werkzeuge und viele darüber hinausgehende Programme wie eine grafische Benutzeroberfläche (GUI), Arbeitsumgebungen wie z.B. GNOME oder KDE, Anwendungssoftware wie z.B. OpenOffice oder GIMP, oder die bekannten Serverdienste Apache und MySQL. Diese Distributionen werden sowohl von Einzelpersonen, Gruppen und gemeinnützigen Organisationen, als auch von professionellen Distributoren erstellt. Das Resultat wird meist Linux genannt, aber auch GNU/Linux, um dem GNU Projekt der Free Software Foundation für seinen Beitrag der GNU-System-Software für seinen Beitrag der GNU System Software Anerkennung zu zollen. Distributionen gibt es wie Sand am Meer. Vom Einsteigersystem mit automatisierter Konfiguration bis zum Hardcore-Linux ist alles zu haben. Die Preise bewegen sich, nach heutigem Standart, zwischen 20,- und höchstens 100,- Euro. Diesen Betrag verlangen die Distributoren nur für das Zusammenstellen der Software, die bekanntlich unter der GNU-License steht.

Linux läuft z.Zt. auf folgenden Architekturen:

- Acorn Archimedes, A5000 und RiscPC Serie (ARM, StrongARM, Intel XScale etc.)
- AMD64 (auch bekannt als x86-64): AMDs 64-bit Prozessoren Athlon 64 und Opteron sowie Intel Prozessoren mit EM64T-Erweiterung (Xeon 4E)
- Axis Communications' CRIS
- Compaq Alpha-Prozessor
- Hitachi H8/300
- Hewlett Packard PA-RISC
- IA-64: PCs mit 64bit Intel Itanium-Prozessor
- IBM S/390
- Intel 80386 und neuer: IBM-PCs und kompatible mit den CPUs 80386, 80486, und Pentium-Serie; AMD Athlon, Duron, Thunderbird; Cyrix-Prozessoren.
- MIPS: Maschinen von Silicon Graphics
- Motorola 68020 und neuer: neuere Amigas, einige Apple Computer
- NEC v850e
- PowerPC: Alle neueren Apple Computer, Clones der Power Macs von Power Computing, UMAX und Motorola, mit einer „Power-UP"-Karte verbesserte Amigas, POWER und PowerPC-basierte IBM RS/6000-Systeme, verschiedenen embedded PowerPC-Plattformen
- Sun SPARC und UltraSparc: Sun-Workstations
- Hitachi SuperH: Sega Dreamcast
- Sony PlayStation 2
- Microsofts Xbox
- Nintendos GameCube Link



Die Linux-Verzeichnisstruktur

Bei der Unmenge von Dateien in Unix-Systemen ist eine hierarchische Struktur unabdingbar. Lange Zeit brachte jedes UNIX Derivat seine eigenen Vorstellungen vom Aufbau seiner Dateiverwaltung mit, aber unterdessen ist sich ein Gremium, bestehend aus den wichtigsten Linux Distributoren, System- und Paketentwicklern, mehr oder weniger einig geworden und erarbeitete den Filesystem Hierarchie Standard, der wichtige Strukturen definiert. Die meisten Distributionen folgen diesen Richtlinien, wobei Abweichungen oft durch unterschiedliche Auslegung des Standards begründet werden.

Neben der Beschreibung der vorgesehenen Verwendung jedes Verzeichnisses werden auch konkrete Kommandos genannt, die mindestens in diesen Verzeichnissen vorhanden sein müssen. Des Weiteren finden hier systemspezifische Vorgaben und optionale Komponenten kurz Erwähnung.

Ausgangspunkt ist die Wurzel (Root).
Sobald der Kernel aktiv ist, lädt er als erstes das Root-Dateisystem, in dem alle für die Aufgaben des Kernels notwendigen Programme und Konfigurationsdateien angesiedelt sein müssen.

Zu den Programmen gehören:

- Dienstprogramme zum Prüfen und Reparieren des Dateisystems
- Programme zum Sichern der Systemdaten und zur Installation neuer Systemteile
- Eventuell wichtige Netzwerkprogramme

Verzeichnisse in der Wurzel
Wenden wir uns also zunächst dem Inhalt des Wurzelverzeichnisses zu.

/bin

Die wichtigsten Kommandos, um mit dem System arbeiten zu können, sind hier aufgelistet. Sie dürfen von jedem Benutzer ausgeführt werden. Zu den Kommandos gehören *cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo, ed, false, kill, ln, login, ls, mkdir, more, mount, mv, ps, pwd, rm, rmdir, sed, setserial, sh, stty, su, sync, true, umount, uname* (sh ist in Linuxsystemen zumeist ein Link auf bash).

/boot

Hier sind die statischen Dateien des Bootmanagers und die Kernel abgelegt (In früheren Linux-Versionen wurden der »Haupt-Kernel im Root-Verzeichnis installiert und nur die optionalen Kernel in diesem Verzeichnis).

/dev

In diesem Verzeichnis stehen die Treiber zur Ansteuerung der gesamten Hardware (Festplatte, Floppy, RAM...). Gemäß der UNIX-Philosophie »Alles ist eine Datei« werden diese Treiber durch Dateien repräsentiert, die so genannten Gerätedateien (»device special files« oder »device nodes«). Einige dieser Gerätedateien erfüllen spezielle Aufgaben (Beispiele finden Sie an etlichen Stellen des Buches) und sind mit keiner realen Hardware verbunden. Dennoch stellen sie aus Sicht des Systems Treiber dar.

Wenn man sich den Inhalt des Verzeichnisses »/dev« betrachtet, entdeckt man unzählige Einträge, deren Anzahl unmöglich der vorhandenen Hardware entsprechen kann. Tatsächlich wurde für jedes denkbare Stück Hardware vorsorglich ein solcher »Eintrittspunkt« geschaffen, dessen definierter Name es Anwendungsprogrammen erst ermöglicht, auf ein konkretes Gerät zuzugreifen. Solange keine entsprechende Hardware und der zugehörige Treiber installiert wurden, sind diese Eintrittspunkte nur »leere Hülsen« und würde eine Anwendung darauf zugreifen, erhielten sie eine Fehlermeldung, dass das angeforderte Gerät nicht verfügbar ist.

Obige Verfahrensweise beinhaltet jedoch einige Schwächen, auf die wir im einzelnen an dieser Stelle nicht eingehen wollen. Als Lösung bahnt sich seit Kernelversion 2.4. das so genannte Device File System an, womit erst bei der Registrierung eines Treibers beim Kernel eine entsprechende Gerätedatei unterhalb von »/dev« generiert wird. Noch dazu werden Geräte entsprechend ihrer Klasse (bspw. IDE-Festplatten, Terminals, SCSI-Geräte) in separate Unterverzeichnisse eingeordnet, sodass der Inhalt von »/dev« die tatsächliche Hardware des Systems in strukturierter Form repräsentiert.

Noch aktiviert keine Distribution standardmäßig das neue Verfahren und selbst bei dessen Verwendung werden zur Unterstützung älterer Programme die »alten« Gerätedateien zusätzlich verwendet (ein Prozess überwacht hierzu die Anforderungen der Programme und kümmert sich ggf. um das Mapping auf die neuen Einträge unter »/dev«), sodass nachfolgende Aussagen weiterhin ihre Gültigkeit behalten.

Drei Informationen sind für jedes Device relevant:

Art des Zugriffs

```
brw-rw-rw-   1 root     disk       2,   0 Nov  8 20:48 /dev/fd0
crw-rw---    1 root     lp         6,   0 Nov  8 20:48 /dev/lp0
```

- Blockorientiert (b) - gepufferter Zugriff, z.B. Festplatten
- Zeichenorientiert (c) - ungepufferter Zugriff, z.B. Bildschirm, Drucker

Hauptgerätenummer (major device number)

```
brw-rw-rw-   1 root     disk       2,   0 Nov  8 20:48 fd0
crw-rw-rw-   1 root     tty        2,   0 Nov  8 20:48 ptyp0
```

Diese Nummern spezifizieren den zu verwendenden Treiber. Befinden sich im System bspw. zwei IDE-Festplatten, werden beide über ein und denselben Treiber angesprochen. Nicht allein die Nummer benennt den Treiber, sondern die Art des Geräts (zeichen- und blockorientiert) ist ebenso entscheidend. Identische Nummern für zeichen- und blockorientierte Geräte (z.B. Nummer 2 für Pseudoterminals (c) und Floppys (b)) beziehen sich also auf unterschiedliche Treiber!

Die Hauptgerätenummern werden von einer zentralen Instanz verwaltet; der aktuelle Stand bei Auslieferung der Kernelquellen ist in der Datei »/usr/src/linux/Documentation/devices.txt« zu finden.
Nebengerätenummer (minor device number)

```
brw-rw-rw-   1 root     disk       2,   0 Nov  8 20:48 /dev/fd0
```

Diese Nummer dient dem Treiber zu entscheiden, welche Instanz einer Hardware anzusprechen ist. Sind bspw. vier IDE-Festplatten im System installiert, erkennt der Treiber anhand der Nebengerätenummer, ob er den Master oder den Slave des ersten oder zweiten IDE-Controllers anzusteuern hat.

/etc
Enthält alle lokalen Konfigurationsdateien (Tastatur, X, Netzwerk...)

/home
Alle Heimatverzeichnisse der Nutzer finden man standardmäßig hier. Nach dem Login landet jeder Benutzer i.d.R. in seinem »Home«. Heimatverzeichnisse können vom Systemverwalter auch an anderer Stelle angesiedelt werden.

/lib
Die beim Systemstart benötigten Bibliotheken stehen hier. Ebenso liegen die Kernelmodule in einem eigenen Unterverzeichnis unterhalb von /lib

/mnt
Mountpunkt für temporäre Partitionen

/opt
Software, die nicht zum üblichen Installationsumfang von Unix-Systemen gehören, werden oft unter diesem Zweig installiert. So werden nahezu alle kommerziellen Softwarepakete hier eingerichtet; auch die Programme zur KDE befinden sich hier.

/root
Heimatverzeichnis des Administrators. In realen Unix-Installationen werden die Heimatverzeichnisse aller Benutzer oft auf einem Server gehalten. Bei einem Ausfall eines solchen Servers sollte aber zumindest Root in der Lage sein, vernünftig mit dem System zu arbeiten. Daher liegt dessen Heimatverzeichnis direkt unterhalb der Verzeichnisbaumwurzel.

/sbin
Wichtige Systemprogramme (beim Booten benötigt; Ausführung erfordert Root-Rechte)

/tmp
Temporäre Dateien können hier abgelegt werden, jeder Nutzer ist dazu berechtigt.

/usr
Die zweite Hierarchie

/var
Variable Daten

Das System booten

Der Start einer Linux Maschine ist normalerweise einem durchschnittlich intelligenten Menschen binnen weniger Sekunden erklärt: Hauptschalter drücken und bis zur Einlogaufforderung warten.

Es gibt verschiedene Möglichkeiten, Linux auf einem System zu booten. Die gebräuchlichsten sind das Booten von der Festplatte und mittels einer Boot-Diskette. Oft hat die Installationsprozedur bereits eine oder beide dieser Möglichkeiten für konfiguriert. Auf jeden Fall ist es wichtig, dass man versteht, wie der Boot-Vorgang selbst konfiguriert werden kann.

Von einer Diskette booten

Im allgemeinen enthält eine »Boot-Diskette« für Linux einfach eine Kopie des Kernels (kernel image), die beim Booten des Systems in den Arbeitsspeicher geladen wird.

Meistens ist die Kopie des Kernels komprimiert, wobei dieselben Algorithmen benutzt werden, mit denen auch das Komprimierungsprogramm gzip arbeitet (mehr dazu im Abschnitt »Den Kernel kompilieren«). Dadurch ist es möglich, den Kernel, der mehr als ein Megabyte groß sein kann, auf wenige Hundert Kilobytes zu reduzieren. Ein Teil des Kernel-Codes wird nicht komprimiert, dieser Teil enthält die Routinen, die zum Entkomprimieren und Laden der Kernel-Kopie in den Speicher benötigt werden. Der Kernel bereitet sich also beim Booten selbst vor, indem er sich in den Arbeitsspeicher entpackt.

In der Kopie des Kernels sind eine Reihe von Parametern enthalten, u.a. der Gerätename, der beim Booten des Kernels als Root-Dateisystem benutzt werden soll. Ein anderer Parameter bezeichnet den Textmodus für die Systemkonsole. Alle diese Angaben können mit dem Befehl rdev geändert werden, der später in diesem Kapitel vorgestellt wird.

Nach dem Booten des Kernels versucht dieser,

auf dem Gerätenamen, der in der Kernel-Kopie als Root fest einprogrammiert ist, ein Dateisystem aufzusetzen. Dieses Dateisystem wird das Root-Dateisystem sein, d.h., das Dateisystem auf /. Im Augenblick genügt es zu wissen, dass das Kernel-Image den Gerätenamen für das Root Dateisystem enthalten muss. Falls der Kernel auf diesen Namen kein Dateisystem aufsetzen kann, wird er aufgeben und eine »kernel panic« Meldung anzeigen. Eine Kernel-Panik ist ein nicht behebbarer Fehler, der vom Kernel selbst gemeldet wird. Eine Panik wird immer dann auftauchen, wenn der Kernel endgültig durcheinander geraten ist und nicht weiterarbeiten kann. Falls der Kernel z.B. einen Fehler enthält, könnte eine Panik entstehen, wenn er versucht, auf nicht vorhandenen Speicher zuzugreifen.

Der Root-Gerätename in der Kernel Kopie ist der Name des Root Dateisystems auf der Festplatte. Das bedeutet, dass der Kernel beim Booten eine Partition der Festplatte als Root Dateisystem aufsetzt und die Kontrolle dann auf die Festplatte übergeht. Sobald der Kernel in den Speicher geladen wurde, bleibt er dort stehen -- die Boot-Floppy wird dann nicht mehr benötigt (bis zum nächsten Booten).

Viele Linux Distributionen legen bei der Installation des Systems eine solche Boot-Diskette an. Die Boot Floppy bietet eine einfache Methode Linux zu booten, wenn man nicht von der Festplatte booten möchten. (So sind z.B. die Boot Manager von OS/2 und Windows NT nicht ganz problemlos so einzurichten, dass sie Linux starten.)

Sobald der Kernel von der Diskette gebootet hat, kann das Diskettenlaufwerk wieder für andere Zwecke benutzt werden.

Mit einer Kopie des Kernels kann man seine eigene Boot Floppy erstellen. Auf vielen Linux-Systemen wird der Kernel in einer Datei namens /boot/vmlinuz abgelegt. (3)

Dies ist allerdings keine allgemeingültige Regel; andere Linux-Systeme speichern den Kernel in /vmlinuz oder /vmlinux , wieder andere z.B. in /Image. (Falls man mehrere Kernel Kopien hat, können man mittels LILO bestimmen, welcher Kernel gebootet werden soll. Zu LILO später mehr.) Beachten muss man allerdings, dass neu installierte Linux-Systeme eventuell keine Kopie des Kernels auf die Festplatte geschrieben haben wenn eine Boot-Floppy erstellt wurde. Man kann aber auf jeden Fall einen eigenen Kernel erstellen. Dies ist immer eine gute Idee – dabei kann man den Kernel so »anpassen«, dass nur die Treiber für die tatsächlich vorhandene Hardware eingebunden werden.
Los geht's. Nehmen wir an, dass eine Kopie des Kernels in der Datei /boot/vmlinuz steht. Der erste Schritt zur Erstellung einer Boot-Diskette ist, mittels rdev als Root-Gerätenamen den Namen des Linux-Root-Dateisystems anzugeben. (Falls der Kernel selbst erstellt wurde, sollte dies schon geschehen sein, aber es kann nicht schaden, wenn man das mit rdev überprüft.)
Mit dem Befehl rdev -h im Root-Modus bekommt man eine Online-Hilfe angezeigt. Man kann feststellen, dass es eine ganze Reihe von Optionen

gibt, mit denen man die Gerätedateien für Root und Swap, die Größe der RAM-Disk usw. bestimmen kann. Meistens braucht man sich an dieser Stelle mit den anderen Optionen nicht zu befassen.

Mit dem Befehl rdev /boot/vmlinuz , wird das Root-Verzeichnis angezeigt, das im Kernel /boot/vmlinuz einprogrammiert ist:

```
courgette:/# rdev /boot/vmlinuz Root
device /dev/hda1
```

Falls dies nicht korrekt ist und sich das Linux-Root-Dateisystem in Wirklichkeit auf /dev/hda3 befindet, sollten das mit:

```
courgette:/# rdev /boot/vmlinuz /
dev/hda3 courgette:/#
```

geändert werden. rdev gehört zu den eher schweigsamen Befehlen – bei der Festlegung des Root-Verzeichnisses wird keinerlei Meldung ausgegeben man sollte deshalb rdev /boot/vmlinuz noch einmal ausführen, um den letzten Schritt zu überprüfen.

Jetzt ist man soweit, um die Boot-Diskette erstellen zu können. Dazu nimmt man am besten eine neue, formatierte Diskette; sie kann unter DOS oder mit fdformat unter Linux formatiert werden . Mit fdformat werden auch die Informationen zu Spuren und Sektoren auf der Diskette abgelegt, so dass das System die Kapazität der Floppy selbst erkennt.

Mit dem Befehl dd wird das Kernel-Image auf eine Floppy kopiert, um die Boot-Diskette zu erstellen:
courgette:/# dd if=/boot/vmlinuz of=/dev/fd0 bs=8192

Für die, die es interessiert: In der Man-Page findet man eine ausführliche Beschreibung von dd. Hier nur soviel: Man kopiert damit die Eingabedatei (Option if) namens /boot/vmlinuz in die Ausgabedatei (Option of)/dev/fd0 , d.h. das erste Diskettenlaufwerk. Die Blockgröße ist 8192 Bytes. Natürlich ließe sich das auch mit dem Prolo-Befehl cp erledigen, aber wir UNIX-Freaks benutzen lieber kryptische Befehle, auch um relativ einfache Dinge durchzuführen. Das ist der Unterschied zwischen uns und Normalsterblichen!

Damit sollte nun die Boot-Diskette fertig sein. Wenn alles klappt, wird das Linux-System starten wie immer. Vielleicht ist es eine gute Idee, eine zweite Boot-Floppy als Reserve anzulegen.

LILO benutzen

LILO (der Linux Loader) ist ein Boot-Manager, der beliebige Betriebssysteme einschließlich Linux booten kann, die auf einem System installiert sind. Es gibt Dutzende von Möglichkeiten, LILO zu konfigurieren. Wir wollen hier die beiden am häufigsten eingesetzten Methoden beschreiben – die Installation des LILO im Master-Boot-Record (MBR) der Festplatte sowie LILO als sekundärer Boot-Loader nur für Linux.

LILO ist die übliche Methode, Linux von der Festplatte zu booten. Mit »booten von der Festplatte«

meinen wir, dass der Kernel von vornherein auf der Festplatte steht und keine Boot-Floppy benötigt wird. Man sollte sich aber daran erinnern, dass auch beim Booten von einer Diskette die Kontrolle an die Festplatte abgegeben wird, sobald der Kernel in den Speicher geladen wurde. Wenn LILO im Master-Boot-Record (MBR) der Festplatte steht, ist dies der erste Programmcode, der beim Laden der Festplatte ausgeführt wird. LILO ist in der Lage, anschließend andere Betriebssysteme zu booten – etwa Linux oder DOS. Dabei kann während des Boot-Vorgangs entschieden werden, welches Betriebssystem geladen werden soll.

Allerdings benutzen sowohl OS/2 als auch Windows NT ihren eigenen Boot-Manager, der den MBR belegt. Wenn man mit einem dieser Systeme arbeitet, wird man LILO als »sekundären« Boot-Loader speziell für Linux installieren müssen, falls Linux von der Festplatte gestartet werden soll. In diesem Fall wird LILO nur im Boot-Record der Linux-Root-Partition installiert und die Boot-Manager von OS/2 oder NT rufen dann LILO auf, wenn Linux gebootet werden soll.

Wir werden allerdings noch feststellen, dass die Boot-Manager von OS/2 und NT sich irgendwie unkooperativ verhalten, wenn sie LILO aufrufen sollen. Dies wäre deshalb eine ungünstige Systemkonfiguration. Wenn man also unbedingt einen dieser beiden Boot-Manager benutzen möchte, könnte es einfacher sein, Linux von einer Diskette zu starten.
Bevor wir weitermachen, sollten wir noch darauf hinweisen, dass einige Linux-Distributionen beim Einspielen der Software auch LILO installieren und konfigurieren. Manchmal ist es allerdings günstiger, LILO selbst zu konfigurieren, damit wirklich alles korrekt eingerichtet wird.

Die Datei /etc/lilo.conf

Der erste Schritt zur Einrichtung von LILO ist das Erstellen der Konfigurationsdatei, die oft /etc/lilo.conf heißt. Auf manchen Systemen kann das auch /boot/lilo.conf oder /etc/lilo/config sein.

Wir werden uns eine beispielhafte lilo.conf -Datei Zeile für Zeile ansehen. Man kann diese Datei als Grundlage für eine eigene lilo.conf benutzen und sie an das System anpassen.

Im ersten Teil der Datei werden einige grundlegende Parameter gesetzt.
boot = /dev/hda
compact
install = /boot/boot.b
map = /boot/map

Die Boot-Zeile gibt den Gerätenamen an, in dessen Boot-Record LILO sich selbst installieren soll. In diesem Beispiel wollen wir LILO im Master-Boot-Record von /dev/hda einrichten, der ersten Nicht-SCSI-Festplatte. Falls man von einer SCSI-Platte booten will, gibt man statt dessen einen Gerätenamen wie etwa /dev/sda an. Wenn man den Namen einer Partition (z.B. /dev/hda2) statt des Namens einer Festplatte angibt, wird LILO als sekundärer Boot-Loader in der angegebenen Partition installiert.

Die Zeile compact lässt LILO einige Optimierungen durchführen; diese Zeile sollte immer angegeben sein, es sei denn, man will ernsthaft mit der Konfiguration von LILO experimentieren. Ebenso sollten auf jeden Fall die install- und map-Zeilen angegeben werden. Install bezeichnet die Datei, die den Boot-Sektor enthält, der im MBR benutzt werden soll. Map gibt die »Map-Datei« an, die LILO bei der Installation erstellt. Diese Dateien sollten im Verzeichnis /boot stehen, obwohl sie auf einigen Systemen auch unter /etc/lilo zu finden sind. Die Datei /boot/map wird erst bei der Installation von LILO angelegt.

Anschließend fügt man für jedes Betriebssystem, das LILO booten soll, noch eine »Strophe« in /etc/lilo.conf ein. Eine Linux-Strophe könnte z.B. so aussehen:

```
# Strophe für Linux mit Root-Partition auf /dev/hda2.

image = /boot/vmlinuz    # Lage des Kernels

    label = linux    # Name des Betriebssystems (für

LILO-Boot-Menü)

   root = /dev/hda2      # Lage der Root-Partition

   vga = ask    # VGA-Text-Modus beim Booten abfragen
```

Die image-Zeile gibt den Namen der Kernel-Kopie an. Untergeordnete Felder sind z.B. label, das diese Strophe mit einem Namen versieht, der auch im Boot-Menü von LILO erscheint; root benennt die Root-Partition von Linux; und vga bezeichnet den zu wählenden Textmodus für die Systemkonsole.

Gültige Einträge in der vga-Zeile sind normal (für normale 80x25-Bildschirme), extended (für erweiterte Textmodi wie 132x44 oder 132x60), ask (bewirkt, dass der Textmodus beim Booten abgefragt wird) oder ein Intergerwert (1, 2 usw.). Dieser Wert entspricht der Nummer des Modus, den man bei der Option ask auswählen kann. Es hängt von der Grafikkarte ab, welche Textmodi vorhanden sind; mit vga = ask erhält man eine Liste der Optionen.

Falls man mehrere Linux-Kernel hat, die man booten möchten – z.B. zum Debuggen des Kernels – kann man für jeden Kernel eine image-Strophe einfügen. Nur der Unterpunkt label muss vorhanden sein. Wenn man root oder vga nicht angibt, werden die Voreinstellungen benutzt, die mittels rdev in der Kernel-Kopie selbst abgelegt wurden. Falls man root oder vga angibt, haben diese Angaben Vorrang vor denen, die eventuell mit rdev eingestellt wurden. Wenn man Linux unter Benutzung von LILO startet, besteht deshalb keine Notwendigkeit mehr, rdev zu benutzen; die LILO-Konfiguration setzt diese Parameter.

Eine Strophe für das Booten von MS-DOS würde so aussehen:

```
# Strophe für MS-DOS-Partition auf
```

```
/dev/hda1.

other = /dev/hda1    # Lage der Par-
tition

   table = /dev/hda  # Lage der Par-
titionstabelle für /dev/hda2

   label = msdos     # Name des Be-
triebssystems (für das Boot-Menü)
```

Mit einer ähnlichen Strophe kann man auch OS/2 von LILO aus booten (die label-Zeile muss natürlich anders lauten).

Wenn man ein DOS booten will, das auf der zweiten Festplatte steht, sollten man die Zeile:

```
loader = /boot/any_d.b
```

im Punkt other der DOS-Strophe einfügen. Bei OS/2-Partitionen auf der zweiten Festplatte gibt Sie ein:

```
loader = /boot/os2_d.b
```

Es gibt viele weitere Optionen zur Konfiguration von LILO. Die LILO-Distribution selbst (gibt es bei den meisten Linux-FTP-Servern und -Distributionen) enthält eine ausführliche Beschreibung aller Optionen. Die hier besprochenen Hinweise sollten allerdings für die meisten Systeme ausreichen.

Nachdem man die /etc/lilo.conf erstellt hat, kann man als root den Befehl:

```
/sbin/lilo
```

ausführen lassen. Danach sollte etwa folgende Information angezeigt werden:

```
courgette:/# /sbin/lilo Added linux
Added msdos
courgette:/#
```

Mit dem Schalter -v gibt lilo weitere Informationen aus, die hilfreich sind, wenn etwas nicht geklappt hat. Die Option -C erlaubt die Benutzung einer anderen Konfigurationsdatei als /etc/lilo.conf .

Nach diesem Schritt ist man soweit, das System herunterfahren zu können, um es anschließend neu zu starten und weiter zu erforschen. Als Voreinstellung werden die Befehle in der Strophe zum ersten Betriebssystem in /etc/lilo.conf ausgeführt. Wenn man einen anderen Kernel oder ein anderes Betriebssystem aus /etc/lilo.conf auswählen möchten, drückt man die SHIFT-, ALT- oder CTRL-Taste, während das System bootet. Das sollte den LILO-Boot-Prompt auf den Bildschirm bringen:
boot:

An dieser Stelle erhält man durch Drücken von TAB eine Liste der vorhandenen Boot-Optionen:
boot: tab-taste linux msdos

Dies sind die Namen, die in den label-Zeilen in /etc/lilo.conf angegeben wurden. Gibt man den Namen des entsprechenden Labels ein, wird dieses Betriebssystem gestartet. In unserem Beispiel würde nach der Eingabe von msdos von /dev /hda1 aus MS-DOS gebootet – wie wir es in der Datei lilo.conf vorgesehen hatten.

LILO als sekundärer Boot-Loader

Wenn man den Boot-Manager von OS/2 oder NT benutzt oder nicht möchte, dass LILO den Master-Boot-Record der Festplatte belegt, kann man LILO als sekundären Boot-Loader konfigurieren, der nur im Boot-Record der Linux-Partition steht.

Dazu ändert man einfach die Zeile boot = ... in /etc/lilo.conf so, dass sie den Namen der Linux-Root-Partition enthält. Ein Beispiel:
boot = /dev/hda2 installiert LILO im Boot-Record von /dev/hda2 , so dass nur Linux gebootet wird. Dies passiert aber nur in primären Partitionen (also nicht in erweiterten oder logischen Partitionen).

Damit Linux auf diesem Wege gebootet werden kann, sollte seine Root-Partition in der Partitionstabelle als »active« markiert werden. Das kann mit fdisk unter Linux oder DOS erledigt werden. Beim Booten des Systems wird das BIOS den Boot-Record der »aktiven« Partition lesen und Linux starten.

Wenn man den Boot-Manager von OS/2 oder Windows NT benutzen, sollte man LILO auf diese Weise installieren. Dann muss dem Boot-Manager mitgeteilt werden, dass er ein weiteres Betriebssystem aus dieser Partition der Festplatte heraus booten kann. Es hängt vom betreffenden Boot-Manager ab, wie man ihm das mitteilt.

Obwohl LILO mit dem Boot-Manager von OS/2 zusammenarbeitet, ist es manchmal schwierig, alles korrekt einzurichten. Das Problem liegt darin, dass der Boot-Manager von OS/2 die Partitionen nicht erkennt, die man mit dem fdisk von Linux angelegt hat. Der Ausweg aus dieser Situation ist das Einrichten der Linux-Partitionen mit dem fdisk von OS/2 mit anschließender Formatierung z.B. als Partition mit einer DOS-FAT.

OS/2 wird diese Partitionen erkennen und man kann anschließend mit dem fdisk von Linux die Partitionstypen Linux native und Linux swap einstellen. Anschließend kann man Linux in diesen Partitionen installieren und dabei LILO im Boot-Record der Linux-Root-Partition unterbringen; dann wird (hoffentlich) alles funktionieren.

Warum erzählen wir das jetzt? Weil der Boot-Manager von OS/2 nicht korrekt funktioniert, wenn es um das Booten von ihm unbekannten Betriebssystemen geht. Statt des OS/2-Boot-Managers kann man aber auch LILO im MBR der Festplatte installieren und auf diesem Wege auch OS/2 booten. Dazu fügt man für OS/2 eine weitere other...-Strophe in der Datei /etc/lilo.conf ein, wie man es auch für MS-DOS tut. Eine andere Möglichkeit ist es, Linux einfach von einer Diskette zu starten – oder, noch besser, auf Microsoft zu verzichten.

Die Boot-Optionen festlegen

Die Wahrscheinlichkeit ist groß, dass man bei der Linux-Installation von einer Diskette gebootet hat und dann den inzwischen bekannten Boot-Prompt von LILO sieht.

An diesem Prompt kann man eine Reihe von Boot-Optionen eingeben, z.B.:
hd= zylinder ,köpfe ,sektoren

Damit gibt man dem System die Geometrie der Festplatte bekannt. Es kann sein, dass man bei jedem Booten diese Parameter angeben muss, damit die Hardware korrekt erkannt wird, wie wir das im Abschnitt »Linux booten« beschrieben haben.

Wenn man LILO benutzen, um Linux von der Festplatte zu booten, kann man diese Parameter auch in /etc/lilo.conf spezifizieren, statt sie immer wieder am Boot-Prompt einzugeben. Dazu fügt man in lilo.conf im Linux-Abschnitt eine Zeile wie:
append = „hd=683,16,38" ein. Damit bewirkt man, dass sich das System gerade so verhält, als ob man hd=683,16,38 am LILO-Boot-Prompt eingegeben hätte. Man kann in einer append Zeile mehrere Boot-Optionen angeben, etwa so:
append = „hd=683,16,38 hd=64,32,202"

In diesem Beispiel haben wir die Geometrien für die erste bzw. zweite Festplatte bekanntgegeben.

Man muss beachten, dass solche Boot Optionen nur dann angeben werden müssen, wenn der Kernel der Hardware beim Booten nicht korrekt erkannt wird.

Es gibt noch einige weitere Boot-Optionen. Die meisten werden zur Erkennung von Hardware benötigt, die wir bereits besprochen haben.

Hier noch einige ganz nützliche Optionen:

single
Das System startet im Single-User-Modus (Ein-Benutzer-Modus); die Systemkonfiguration wird übersprungen und es wird eine Root-Shell auf der Konsole gestartet.

root=partition
Versucht, die angegebene Partition als Linux-Root-Dateisystem aufzusetzen. Diese Option hat Vorrang vor den Einträgen in /etc/lilo.conf .

ro
Setzt das Root-Dateisystem als read-only (nur lesbar) auf. Dies geschieht in der Regel vor der Ausführung von fsck.

ramdisk=größe
Gibt die Größe der RAM-Disk in Bytes an. Hat Vorrang vor Einträgen in /etc /lilo.conf . Die meisten Benutzer brauchen sich über die Benutzung der RAM-Disk keine Gedanken zu machen, da Sie in erster Linie bei der Installation benötigt wird.

vga=modus
Bestimmt den VGA-Modus. Dies hat Vorrang vor Einträgen in /etc/lilo.conf . Gültige Modi sind normal, extended, ask oder ein Integerwert. Diese Werte entsprechen den vga = -Werten, die in lilo.conf eingetragen werden.

Diese Optionen können alle am LILO-Boot-Prompt von Hand eingegeben oder auch mit der Option

append in /etc/lilo.conf eingetragen werden.
LILO wird mit vollständiger Dokumentation ausgeliefert, in der alle Optionen beschrieben sind. Auf vielen Linux-Systemen findet sich diese Dokumentation in /usr/src/lilo . Falls man überhaupt keine Beschreibung auftreiben kann, sollten man sich die LILO-Distribution von einem der Linux-Server holen oder den Händler bitten, den Quellcode und die Dokumentation von LILO zu besorgen. Zur Dokumentation gehören ein Handbuch, in dem die Konzepte rund um LILO im Detail beschrieben werden sowie eine README -Datei mit Ausschnitten aus diesem Handbuch in Form von ASCII-Text.

LILO entfernen

Wenn man LILO im Master-Boot-Record installiert hat, führt der schnellste Weg zum Entfernen über den Befehl FDISK von DOS. Mit:
FDISK /MBR
Dazu startet man FDISK und überschreiben den MBR mit einem gültigen DOS-Boot-Record.

LILO speichert Sicherungskopien des Original-Boot-Records in den Dateien /boot/boot.0300 (bei IDE-Laufwerken) und /boot/boot.0800 (bei SCSI-Laufwerken). Diese Dateien enthalten den MBR im Zustand vor der Installation von LILO. Mit dd kann man den Boot-Record auf der Festplatte durch diese Kopie ersetzen. Der Befehl:
dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1

kopiert die ersten 446 Bytes der Datei /boot/boot.0300 nach /dev/hda . Obwohl die Dateien 512 Bytes lang sind, sollten nur die ersten 446 Bytes auf den MBR zurückkopiert werden.

Bei diesem Befehl ist größte Vorsicht geboten! Dies ist eine der Stellen, an denen das gedankenlose Eintippen von Befehlen aus einem Buch wirklich Schaden anrichten kann, wenn man nicht weiß, was man eigentlich tut. Diese Methode sollte nur als letzter Ausweg angewendet werden – und nur wenn es sicher ist, dass /boot/boot.0300 oder /boot/boot.0800 genau den Boot-Record enthält, den man braucht. Viele Linux-Distributionen enthalten unbrauchbare Versionen dieser Dateien; vielleicht ist es eine gute Idee, sie vor der Installation von LILO zu löschen.

Fußnoten

(2)
Unter Linux kann eine Boot-Floppy auch einen LILO-Boot-Record enthalten, der dann das System von der Festplatte startet. Wir werden im nächsten Abschnitt darauf eingehen, wenn wir LILO besprechen.
(3)
Woher stammt der merkwürdige Name? Auf vielen UNIX-Systemen steht der Kernel in der Datei /vmunix. Natürlich muss Linux wieder anders sein: deshalb nennt man seine Kernel-Kopien vmlinux und speichert sie im Verzeichnis /boot, um sie aus dem Root-Verzeichnis herauszuhalten. Der Name vmlinuz wurde gewählt, um komprimierte Kernel-Kopien von den unkomprimierten zu unterscheiden. Eigentlich spielen Name und Speicherort des Kernels absolut keine Rolle, solange Sie entweder eine Boot-Floppy mit Kernel benutzen oder aber

LILO mitteilen, wo er ein Kernel-Image findet.

Systemstart und -initialisierung

In diesem Abschnitt werden wir besprechen, was genau beim Booten des Systems passiert. Zur Durchführung einiger Systemkonfigurationen ist das Verständnis dieser Vorgänge und der beteiligten Dateien unerläßlich.

Boot-Meldungen des Kernels

Der erste Schritt ist das Booten des Kernels. Wie wir im vorhergehenden Abschnitt gezeigt haben, kann dies von einer Diskette oder von der Festplatte geschehen. Während der Kernel in den Arbeitsspeicher geladen wird, erscheinen auf der Systemkonsole Meldungen wie:

```
20480 bytes for swap cache allocated
Console: colour EGA+ 80x25, 8 virtual con-
soles
Serial driver version 4.00 with no serial op-
tions enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
tty03 at 0x02e8 (irq = 3) is a 16550A
snd2 <SoundBlaster Pro 3.2> at 0x220 irq 5
drq 1
snd1 <Yamaha OPL-3 FM> at 0x388 irq 0 drq 0
Drive 0: CR-52x-x (2.11)
SBPCD: 1 SoundBlaster CD-ROM drive(s) at
0x0230.
Calibrating delay loop.. ok - 33.22 BogoMips
Memory: 19312k/20480k available (476k kernel
code, 384k reserved, 308k data)
This processor honours the WP bit even when
in supervisor mode. Good.
Floppy drive(s): fd0 is 1.44M
Swansea University Computer Society NET3.016
Checking 386/387 coupling... Ok, fpu using
exception 16 error reporting.
Linux version 1.1.37 (root@loomer) #4 Fri Sep
30 00:08:33 EDT 1994
Partition check:
  hda: Maxtor 7213 AT (202MB IDE w/64KB Cache,
MaxMult=32, CHS=683/16/38)
  hda: hda1 hda2
  hdb: WDC AC140M (40MB IDE w/31KB Cache, Ma-
xMult=8, CHS=977/5/17)
  hdb: hdb1 hdb2
```

Es ist der Kernel selbst, der diese Meldungen ausgibt, wenn die Gerätetreiber initialisiert werden. Welche Meldungen ausgegeben werden, hängt davon ab, welche Treiber in dem jeweiligen Kernel einkompiliert sind und welche Hardware in das System eingebunden ist. Im folgenden beschreiben wir, was die einzelnen Meldungen bedeuten.

Zunächst sehen wir, dass der Kernel 20 Kilobytes als »Swap-Cache« zugewiesen hat. Dieser Cache dient dem beschleunigten Auslagern und Zurücklesen von Speicherseiten zwischen Arbeitsspeicher und Festplatte (mehr dazu später). Danach wird der Typ der Systemkonsole angezeigt; dies betrifft aber nur den Textmodus , den der Kernel benutzt, nicht aber die Möglichkeiten der Grafikkarte. Auch eine SVGA-Karte wird als EGA+ gemeldet, soweit der Textmodus der Konsole betroffen ist.

Als nächstes wird der Treiber für die seriellen Schnittstellen initialisiert, der für jede gefundene Schnittstelle einige Informationen ausgibt. Eine Zeile wie:
tty00 at 0x03f8 (irq = 4) is a 16450 besagt, dass die erste serielle Schnittstelle (/dev/tty00 oder COM1) unter der Adresse 0x03f8 gefunden wurde, IRQ 4 benutzt und einen UART 16450 enthält.

Danach wird der Treiber für die Soundkarte initialisiert; die Namen und Adressen der verschiedenen Soundperipheriegeräte werden angezeigt. In diesem Fall wird eine Soundblaster-Pro-Karte zusammen mit einem Soundblaster Pro-CD-ROM-Laufwerk gefunden.

Die nächste Meldung zeigt die Berechnung der »BogoMips« für Ihren Prozessor. Es handelt sich um eine absolut falsche Berechnung (bogus: Schwindel – daher der Name), die zur optimalen Gestaltung der Warteschleifen in einigen Gerätetreibern benutzt wird. Der Kernel gibt außerdem Informationen zum Arbeitsspeicher aus:
Memory: 19312k/20480k available (476k kernel code, 384k reserved, 308k data)

In diesem Beispiel stehen dem System 19.312 Kilobytes an RAM zur Verfügung. Demnach belegt der Kernel selbst 1168 Kilobytes.

Anschließend wird der Treiber für das Diskettenlaufwerk initialisiert. Das System in unserem Beispiel hat keine Netzwerkkarte installiert; bei vorhandener Ethernet-Karte werden weitere Informationen angezeigt.

Zum Abschluß wird der Treiber für IDE-Festplatten initialisiert, und für jedes IDE-Laufwerk werden der Typ und eine Liste der vorhandenen Partitionen angezeigt.
Abhängig von der Hardware werden eventuell noch weitere Meldungen ausgegeben. So werden zu diesem Zeitpunkt auch parallele Schnittstellen und SCSI-Treiber initialisiert, falls vorhanden.

Diese Meldungen werden beim Booten auf der Konsole ausgegeben und in der Regel auch in den System-Logdateien mitgeschrieben.

init und inittab

Nach der Initialisierung der Gerätetreiber führt der Kernel das Programm init aus, das in einem der Verzeichnisse /etc ,/bin oder /sbin steht (auf den meisten Systemen in /sbin/init).

init ist ein Programm mit mehreren Aufgaben, das neue Prozesse erzeugt und bestimmte Programme nach deren Terminierung wieder startet. So läuft z.B. auf jeder virtuellen Konsole ein getty -Prozeß, der von init gestartet wird. Wenn Sie an einer der virtuellen Konsolen eine Login-Sitzung beenden, terminiert der getty -Prozeß und init startet einen neuen, damit Sie wieder einloggen können.

init ruft außerdem beim Booten des Systems eine Reihe von Programmen und Skripts auf. Die Datei /etc/inittab kontrolliert sämtliche Aktionen von init . Jede Zeile dieser Datei hat das Format:
code :runlevel :aktion :befehl

Code ist ein eindeutiger Name aus einem oder zwei Buchstaben, der jeden Eintrag in dieser Datei identifiziert. Einige dieser Einträge müssen einen bestimmten Code haben, damit sie korrekt funktionieren.

Runlevel

Linux kennt verschiedene Modis in denen es läuft. Dies Modis werden Runlevel genannt. Runlevel ist eine Liste der »Runlevels«, auf denen der betreffende Eintrag ausgeführt werden soll. Ein Runlevel ist einfach eine Ziffer oder ein Buchstabe zur Bezeichnung des aktuellen Systemzustands bei der Ausführung von init . Die Runlevel sind vergleichbar mir der Gangschaltung eines Autos. Man schaltet beim Anfahren nacheinander die Gänge hoch. Zum Halten schaltet man wieder runter. Nicht bei jeder Fahrt werden alle Gänge benötigt.
Beim Hochfahren des Systems werden nacheinander die Runlevels durchlaufen. Es gibt große Abweichungen der Runlevel bei den unterschiedlichen Linux Versionen.

Runlevels bei System V:

Runlevel	Bedeutung
0	Halt
S	Single User Mode
1	Single User Mode ohne Netzwerk
2	Single User Mode mit Netzwerk
3	Multi User Mode mit Netzwerk
4	Frei
5	Multi User Mode mit Netzwerk und X
6	Reboot

Beim Runlevel Halt ist das System heruntergefahren. Der Single User Mode dient zum Pflegen des Systems. Wie der Name schon sagt, kann nur ein Nutzer in diesem Mode mit dem System arbeiten. Dies ist bei Wartungsarbeiten unbedingt notwendig, da andere User mit ihren Prozessen diese Arbeit stören können. Runlevel 1 ermöglicht hier mehrere Nutzer die Arbeit mit dem System. Bei Level 2 & 3 wird das Netzwerk bzw. das graphische Login dazugeschaltet.

Ein Beispiel: Wenn der Runlevel des Systems auf 3 geändert wird, werden diejenigen Einträge in /etc/inittab ausgeführt, die im Feld Runlevel eine 3 stehen haben. Die Runlevels bieten eine einfache Methode, die Einträge in /etc/inittab gruppenweise zusammenzufassen. Sie könnten z.B. festlegen, dass Runlevel 1 nur das notwendige Minimum an Skripts ausführt, Runlevel 2 alles aus Runlevel 1 und zusätzlich die Netzwerkkonfiguration durchführt, während Runlevel 3 alles aus Runlevel 1 und 2 sowie den Wählzugang zum System konfiguriert usw.

Definition der Runlevel

Die Definition der Runlevel erfolgt in der Datei /etc/inittab/ . Eine zerstörte /etc/inittab/ kann dazu führen, dass das System nicht mehr richtig hochgefahren wird. Solange man aber Linux von der Diskette starten gibt es damit keine Probleme.

```
/#          less          /etc/inittab
id:4:initdefault:
r:S:sysinit:/etc/rc.4
l:1:once:/etc/rc.1
f:4:wait:/etc/rc.4
x:5:wait:/etc/rc.5
ca::ctrlaltdel:/etc/rc.6
10:0:wait:/etc/rc.0
16:6:wait:/etc/rc.6
c1:45:respawn:/bin/agetty 38400 tty1
```

```
c2:45:respawn:/bin/agetty 38400 tty2
c3:45:respawn:/bin/agetty 38400 tty3
c4:45:respawn:/bin/agetty 38400 tty4
c5:45:respawn:/bin/agetty 38400 tty5
c6:45:respawn:/bin/agetty 38400 tty6

#-----------------------------------
----------
# servers & deamons
#-----------------------------------
----------

# at command scheduler
AT:5:respawn:/usr/bin/atd
```

Jede Zeile die nicht mit # beginnt ist ein Datensatz mit

In der Regel brauchen muss man sich mit den Runlevels nicht weiter zu befassen. Das System begibt sich beim Booten auf den (in /etc/inittab) voreingestellten Runlevel. Bei den meisten Systemen ist das Runlevel 5. Solange man keinen besonderen Grund hat, das System nicht mit allen hier beschriebenen Konfigurationsskripts hochzufahren, kann man einfach den voreingestellten Runlevel benutzen.

Das Feld Befehl enthält den Befehl, den init für diesen Eintrag ausführen wird. Das Feld Aktion weist init an, wie ein Eintrag ausgeführt werden soll, z.B. ob ein Befehl nur einmal ausgeführt oder ob er nach der Terminierung jedesmal erneut aufgerufen werden soll.

Lassen Sie uns einen Blick auf die beispielhafte /etc/inittab in werfen.

Beispiel 4-1. Beispielhafte /etc/inittab-Datei

```
# Set the default runlevel to five
# Voreinstellung für Runlevel ist 5
id:5:initdefault:
# Execute /etc/rc.d/rc.S when the system boots
# Beim Booten des Systems /etc/rc.d/rc.S aufrufen
si:S:sysinit:/etc/rc.d/rc.S
# Run this script when entering runlevel S (single-user mode)
# Bei Runlevel S (Single-User-Modus) dieses Skript ausführen
su:S:wait:/etc/rc.d/rc.K
# Run this script for all other runlevels (multi-user mode)
# Bei allen anderen Runlevels (Multi-User-Modus) dieses Skript ausführen
rc:12345:wait:/etc/rc.d/rc.M
# Executed when we press ctrl-alt-delete
# Wird beim ‚Affengriff' CTRL-ALT-DELETE ausgeführt
ca::ctrlaltdel:/sbin/shutdown -t3 -rf now
# Start agetty for virtual consoles 1
```

```
through 6
# agetty starten auf virtuellen Konsolen
1 bis 6
c1:12345:respawn:/sbin/agetty      38400
tty1
c2:12345:respawn:/sbin/agetty      38400
tty2
c3:45:respawn:/sbin/agetty 38400 tty3
c4:45:respawn:/sbin/agetty 38400 tty4
c5:45:respawn:/sbin/agetty 38400 tty5
c6:45:respawn:/sbin/agetty 38400 tty6
```

Der tatsächliche Inhalt von /etc/inittab hängt von Ihrem System und der Linux-Distribution ab, die Sie installiert haben.

Als erstes wird die Voreinstellung für den Runlevel auf 5 gesetzt. Das Feld Aktion für diesen Eintrag enthält initdefault, was zur Folge hat, dass der angegebene Runlevel zur Voreinstellung wird. Dies ist der Runlevel, der üblicherweise beim Booten des Systems benutzt wird. Sie können diese Voreinstellung überschreiben, indem Sie init mit dem gewünschten Runlevel von Hand aufrufen (z.B. wenn Sie Ihre Konfiguration debuggen). LILO kann auch im Einzelbenutzermodus booten (Runlevel S) – lesen Sie dazu den Abschnitt »Die Boot-Optionen festlegen «.

Der nächste Eintrag weist init an, beim Systemstart das Skript /etc/rc.d/rc.S auszuführen. (Das Feld Aktion enthält sysinit, was besagt, dass dieser Eintrag ausgeführt werden soll, wenn init beim Systemstart aufgerufen wird.) Diese Datei ist einfach ein Shell-Skript, das Befehle zur grundlegenden Systemkonfiguration enthält. Dazu gehören z.B. das Anlegen der Swap-Partition, die Überprüfung und das Aufsetzen der Dateisysteme sowie die Synchronisierung der Systemuhr mit der CMOS-Uhr. Sie können sich diese Datei auf Ihrem System ansehen; wir werden weiter unten in diesem Kapitel noch auf die Befehle darin eingehen. Lesen Sie auch die Abschnitte »Mit Dateisystemen arbeiten «und »Swap-Space benutzen «.

Wie Sie sehen, wird danach das Skript rc.K ausgeführt, wenn der Runlevel S (Single-User-Modus) erreicht wird. rc.K enthält normalerweise Befehle zum Absetzen der Dateisysteme und zum Beenden des Swapping, so dass der Systemverwalter ggf. Wartungsarbeiten durchführen kann. Das Feld Aktion enthält bei diesem Eintrag wait, womit erreicht wird, dass init den betreffenden Befehl ausführt und erst nach der Beendigung dieses Befehls irgendeine andere Aktion startet.

In ähnlicher Weise wird das Skript rc.M für die Runlevels 1 bis 5 durchgeführt – das sind die »Multi-User-Modi«. Wie Sie diese Runlevels voneinander unterscheiden wollen, liegt in Ihrer Hand; auf vielen Systemen gibt es keinen Unterschied zwischen ihnen. rc.M enthält die meisten der verbleibenden Befehle zur Konfiguration des Systems: Dazu gehören der Aufruf des Bildschirmschoners, die Einstellung des Rechnernamens, die Initialisierung des Netzwerks sowie der Start einiger Dämonen wie syslogd (der Log-Dämon für das System) und lpd (der Drucker-Dämon). Auch diese Datei kön-

nen Sie sich ansehen und feststellen, welche Befehle auf Ihrem System aufgerufen werden.

Auf vielen Systemen ruft rc.M weitere Skripts aus /etc/rc.d auf. Das wichtigste darunter ist rc.local , das eine »lokale« Konfiguration durchführt – alle Befehle, die Sie an die Startroutine anhängen möchten, um sich die Arbeit zu erleichtern. Natürlich könnten Sie ebenso gut ein Skript wie rc.M direkt anpassen, aber durch die Benutzung eines anderen Skriptes – rc.local – für solche persönlichen Anpassungen des Systems trennen Sie Ihre Änderungen von der Konfiguration, wie sie in der Original-Distribution enthalten ist. Dies ist wohl eine Frage des persönlichen Geschmacks – technisch gesehen spielt das keine Rolle.

Der nächste Eintrag mit dem Code ca wird ausgeführt, wenn an der Konsole die Tastenkombination CTRL-ALT-DEL (auch bekannt als »Affengriff«) gedrückt wird. Diese Tastenkombination erzeugt einen Interrupt, der normalerweise das System neu starten würde. Unter Linux wird dieser Interrupt abgefangen und an init weitergeleitet, das dann den Eintrag mit dem Aktion -Feld ctrlaltdel ausführt. Der Befehl, der hier ausgeführt wird, nämlich /sbin/shutdown -t3 -rf now , wird das System »sicher« herunterfahren und neu starten. (Lesen Sie auch »Das System herunterfahren «im nächsten Abschnitt.) Auf diese Weise bewahren wir das System vor einem unerwarteten Neustart infolge der Tastenkombination CTRL-ALT-DEL.

Am Ende enthält die Datei inittab eine Reihe von Einträgen, die /sbin/agetty für die ersten sechs virtuellen Konsolen ausführen. agetty ist eine von mehreren getty -Versionen, die es für Linux gibt. getty ist ein Programm, das ein Terminal (etwa eine virtuelle Konsole oder serielle Schnittstelle) öffnet, verschiedene Parameter für den Terminaltreiber setzt und dann /bin/login aufruft, um auf dem Terminal eine Login-Sitzung zu starten. Deshalb muss auf einer virtuellen Konsole, auf der Sie Logins zulassen wollen, ein getty oder agetty laufen. agetty ist die Version, die auf einer ganzen Reihe von Linux-Systemen benutzt wird, während andere mit getty arbeiten, das eine etwas andere Syntax hat. In den Man-Pages finden Sie Details zu getty und agetty auf Ihrem System.

agetty akzeptiert zwei Parameter: eine Baudrate und einen Gerätenamen. Die Schnittstellen für die virtuellen Konsolen unter Linux heißen /dev/tty1 ,/dev/tty2 usw. agetty sucht die Gerätenamen unterhalb von /dev . Die Baudrate sollte für virtuelle Konsolen immer auf 38.400 gesetzt werden.

Beachten Sie, dass im Feld Aktion der agetty -Einträge ein respawn (etwa: Neustart) steht. Dies veranlaßt init , den Befehl in diesem Eintrag immer wieder neu aufzurufen, wenn der agetty -Prozeß terminiert – was bei jedem Ausloggen eines Benutzers der Fall ist.

Inzwischen sollte init eine bekannte Größe darstellen, aber die diversen Dateien und Befehle in /etc/rc.d , die die eigentliche Arbeit erledigen, bleiben rätselhaft. Wir können auf diese Dateien nicht näher eingehen, ohne mehr Hintergrundwissen zu anderen Aufgaben der Systemverwaltung zu

Zugriffsrechte
Ein Passwort für einen Benutzer macht eigentlich nur dann einen Sinn, wenn er seine Dateien vor anderen Benutzern, die sich ja auch im gleichen Dateibaum bewegen können, schützen kann. Um dies zu ermöglichen, sind jeder UNIX Datei bestimmte Zugriffsrechte zugeteilt. Für jede Datei gibt es drei Benutzerklassen:

User - Eigentümer
Benutzer, der diese Datei einrichtet oder welcher nachträglich als Eigentümer eingetragen wurde.

Group - Gruppe
Jeder Benutzer gehört einer Gruppe an. Der Systemverwalter legt Namen und die Zugehörigkeit fest. Eine Gruppe könnte z.B. eine Gruppe von entwicklern eines bestimmten Projekts oder einer bestimmten Abteilung sein.

Others - alle anderen Benutzer
die „restliche Welt" (ohne Eigentümer und Gruppe).

Die drei Benutzerklassen zusammen bezeichnet man als „alle" (all). Für jede Benutzerklasse gibt es nun die folgenden Zugriffsrechte:

- Lesen (read)
- Schreiben (write)
- Ausführen (execute)

Für einfache Dateien oder Gerätenamen sind diese Befehle einleuchtend, aber wie werden sie bei Directorys interpretiert?

read
Eine Directory wird in Linux wie eine Datei mit einer besonderen Kennzeichnung behandelt. Wenn nun eine Directory gelesen werden darf, so bedeutet dies, dass der Inhalt dieses Dirctorys am Bildschirm aufgelistet werden kann.

write
Das entsprechende Directory darf modifiziert werden, d.h., dass darin enthaltene Dateien gelöscht und überschrieben werden dürfen; um allerdings überhaupt auf Dateien in diesem Directory zugreifen zu können, was beim Neuanlegen bzw. Löschen notwendig ist, muss zusätzlich auch noch das execute-Recht gegeben sein.

execute
Auf Dateien in einem Directory mit execute-Recht darf zugegriffen werden und es ist erlaubt, einsolches Directoty unter Verwendung des Kommandos cd zum Working-Directory zu machen.

Das Recht, eine Datei zu löschen, ist von den Zugriffsrechten der Datei selbst unabhängig. Wenn ein Benutzer in einem Directory eines anderen Benutzers Schreibrechte besitzt, so kann er dort Dateien löschen, auch wenn diese schreibgeschützt sind.
Es ist üblich ganze Directorys und nicht nur einfache Dateien zu schützen. Dies hat den Vorteil, dass man für Dateien mit sensiblen und privaten Daten eine >>doppelte<< Sicherung einbaut, indem man sie in einem solchen sicheren

Directory unterbringt, denn zu leicht kann das setzen der entsprechenden Zugriffsrechte vergessen werden.

Trotz aller Zugriffsrechte für Dateien gibt es einen besonderen Benutzer: den Super-User. Dieser kann auf alle Dateien zugreifen, auch wenn ihm vom einzelnen Benutzer dazu keine Zugriffsrechte gewährt wurden. Dateien, die für alle anderen Benutzer unlesbar sein sollen, können mit dem Kommando crypt verschlüsselt werden.
Ein Benutzer kann auch Dateien in Directorys anlegen, die nicht von ihm eingerichtet wurden; dazu muss er allerdings in den entsprechenden Directorys Schreibrecht besitzen. Er ist dann Eigentümer dieser Datei, obwohl ihm das übergeordenete Directory selbst nicht gehört.

Zugriffsrechte von Dateien ändern

1. chmod *absolut-modus datei(en)*
   und
2. chmod *symbolischer-modus datei(en)*

Die erste Aufrufform
Für datei(en) können dabei einfache Dateine oder auch Directorys angegeben werden. Für absolut-modus muss ein Oktalwert angegeben werden, der festlegt, welche der 12 Bits des Dateimodus für die angegebenen datei(en) zu setzen bzw. zu löschen sind. Dabei hat jedes einzelne der 12 Bits folgende Bedeutung:

Modus	Bedeutung
0400	Leserecht für Eigentümer
0200	Schreibrecht für Eigentümer
0100	Ausführrecht für Eigentümer
0040	Leserecht für Gruppe
0020	Schreibrecht für Gruppe
0010	Ausführrecht für Gruppe
0004	Leserecht für Andere
0002	Schreibrecht für Andere
0001	Ausführrecht für Andere

Die zweite Aufrufform
Für datei(en) können auch hier einfache Dateien oder Directorys angegeben werden. Für symbolischer-modus gilt folgende Syntax:

[ugoa]operator[rwxslt]

Dabei bedeuten die einzelnen Zeichen:

u	für den Eigentümer (user)
g	für die Gruppe (group)
o	für die anderen Benutzer (others)
a	für alle drei Benutzerklassen (all); entspricht der Angabe ugo. Keine Angabe entspricht auch der Angabe a. Z.B. würde chmod +x ... allen Benutzerklassen Ausführrechte geben

Für operator kann eines der folgenden Zeichen angebene werden:

+	Rechte hinzufügen (relativ)
-	Rechte entfernen (relativ)
=	Rechte als neue Zugriffsrechte vergeben (absolut)

Bei dem angegebenen Zugriffsrechte Muster steht dabei:

r	für Leserechte
w	für Schreibrechte
x	für Ausführrechte
s	für set-user-id (in Zusammenhang mit u)
t	für sticky bit; nur im Zusammenhang mit u wirkungsvoll
l	für exclusiven Lese- und/oder Schreibzugriff

Wenn = verwendet wird, dann muss kein Zugriffsrechte-Muster angegeben sein; fehlendes Zugriffsmuster bedeutet dabei: Entfernen alles entsprechenden Zugriffsrechte.

Um einer Datei datei die Zugriffsrechte s–rwxr-xr– zu geben, könnte einer der folgenden Aufrufe abgegeben werden:

chmod u=rwxs , g=rx , o=r datei

oder

chmod 4754 datei

Erklärung dazu:

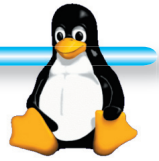
			user			group			others		
s	-	-	r	w	x	r	-	x	r	-	-
1	0	0	1	1	1	1	0	1	1	0	0
4			7			5			4		

Weitere Beispiele zu chmod

Das Kommando passwd, welches sich im Directory /usr/bin befindet, ermöglicht das Ändern des Passwortes. Dieses Kommando muss dazu einen Eintrag in die Datei /etc/shadow machen. Der Besitzer des Kommandes passwd und der Datei /etc/shadow ist root. Damit ein Benutzer nun aber unter Verwendung des Kommandos passwd sein Passwort ändern (in der Datei /etc/shadow schreiben) kann, ist das setuid-Bit für das Kommando passwd gesetzt.

Das setuid-Bits (set-user-id-Bit) ist eine Erfindung des von Dennies Ritchie und dient zur Lösung einer ganzen Reihe von Zugriffsproblemen.
Ein Beispiel dazu: Angenommen, ein Benutzer HANS hat ein Programm KAMPF geschrieben, das Abstimmungen zu gewissen Themenkomplexen erlaubt, wie z.B. gemeinsames Abteilungsessen, Ziel des Betriebsausfluges usw. Dieses Programm ist so ausgelegt, dass es in einem Subdirectory (z.B. abstimmen) von HANS Home -Directory Subdirectorys zu den einzelnen Themenkomplexen erwartet.
Die in diesen Subdirectorys enthaltenen Dateien wahlstand und waehlernamen enthalten den aktuellen Wahlstand und ein eListe der Benutzer, denen die Abgaben einer Stimme zum entsprechenden Thema erlaubt ist; zudem wird in waehlernamen festgehalten, ob ein Benutzer bereits ein Stimme abgegeben hat, um so ein zweimaliges Wählen durch den gleichen Benutzer zu verhindern.

BSD Berkeley Software Distribution



Die Berkely Software Distribution (BSD) ist eine Version des Betriebssystems UNIX, die an der Universität von Kalifornien in Berkeley ab 1977 entstanden ist. BSD basiert auf AT&Ts UNIX Sixth Edition (V6) und Seventh Edition (V7), die ab 1975 bzw. 1979 den Universitäten zum Preis der Datenträger zur Verfügung gestellt wurden, verbunden mit der Erlaubnis, den Quellcode einzusehen und zu modifizieren. Inzwischen ist aber der komplette Quellcode umgeschrieben, und es gibt keine einzige Zeile AT&T-Quellcode in aktuellen BSD Distributionen.

Die ursprüngliche durch die Universität erstellte BSD-Distribution ist kaum noch gebräuchlich. Der Begriff BSD bezeichnet heute viel mehr eine ganze KLasse von Unix-Derivaten, die ihre Wurzel in der eigentlichen BSD haben, wie z.B. die frühen Versionen von SunOS und Ultrix. BSD ist neben System V eine der beiden großen Hauptlinien der UNIX Entwicklung.

Seit den 1990er Jahren sind aus BSD die freien Betriebssysteme FreeBSD (1993), NetBSD (1993), OpenBSD (1995) und DragonFlyBSD (2003) entstanden. Ein auf BSD basierendes Betriebssystem (und mit FreeBSD recht verwandt) wurde als BSD/OS von BSDi, später von Wind River Systems, Alameda, Kalifornien kommerziell vertrieben.

Die BSD-Lizenz

Der Quellcode der Distribution wurde von Berkeley unter der BSD-Lizenz freigegeben, eine freie Lizenz, die heute in modifizierter Form auch für andere Programmpakete verwendet wird. Im Gegensatz zur GNU General Public License erlaubt es die BSD-Lizenz, unter Beachtung einiger Regeln, den Quellcode zur Entwicklung eigener, proprietärer Programme zu verwenden. Diese Programme können dann unter eine beliebige andere (restriktivere) Lizenz gestellt werden, wie es Apple bei der Entwicklung von Mac OS X und dessen Basis Darwin getan hat.

Neuerungen von Berkeley

BSD hatte großen Einfluss auf die Unix-Entwicklung. Es wurden Änderungen am Kernel vorgenommen, das System wurde auch sonst wesentlich erweitert. Viele dieser Neuerungen wurden später entweder direkt oder in ähnlicher Form auch in die konkurrierende System-V-Linie übernommen.

Die wichtigsten Neuerungen waren:

- Das BSD-Fast Filesystem, ein schnelleres Dateisystem mit langen Dateinamen (die 7er-Edition erlaubte nur 14 Zeichen)
- Die TCP/IP Netzwerkimplementierung
- Die Socket-Schnittstelle, eine allgemeine Netzwerk-Programmierschnittstelle, die auf anderen Systemen übernommen wurde.
- Die virtuelle Speicherverwaltung wurde un ter UNIX in Berkeley realisiert.
- Neue Kommandos, die C-Shell csh, der Edtor vi, sowie unter anderem rsh, rexec und rcp.
- Neue Dämonen wie z. B. timed und sendmail.

Die Anfänge

Die Universität Berkeley erhielt 1974 von AT&T die 4er-Edition des noch neuen Betriebssystems UNIX. Dieses war gerade in C umgeschrieben worden, ein C-Entwicklungssystem war Teil des Systems. Sofort wurde mit der Erweiterung des Systems, das auf einer PDP-11 Maschine der Firma Digital Equipment Corporation (Digital) lief, begonnen. Die Arbeiten beschleunigten sich, als 1976 Ken Thompson, ein maßgeblicher Unix-Entwickler, eine Gastprofessur in Berkeley antrat. Inzwischen waren auch die 5er- und 6er-Edition erschienen.

Die Erweiterungen und Änderungen stellte der damalige Student Bill Joy 1977 erstmals zusammen und stellte es externen Interessenten auf einem Magnetband zur Verfügung – die erste Berkeley-Software-Distribution. 1978 waren bereits so viele neue Teile hinzugekommen, dass die zweite Berkeley Software Distribution (2BSD) zusammengestellt wurde. Diese enthielt bereits die erste Version des Editors vi.

VAX Computer und DARPA Einfluss

Inzwischen hatte Digital eine neue Prozessorlinie namens VAX herausgebracht, deren erster Vertreter die VAX-11/780 war. VAX hieß eigentlich Virtual Address Extension und war zunächst als Erweiterung für die PDP-11 Linie gedacht. Die VAX wurde aber eine eigene Prozessorlinie und Digital plante, den Verkauf von PDP-Maschinen zugunsten der VAX-Linie einzustellen. AT&T hatte zwar bereits eine Portierung von UNIX V7 auf den VAX-Prozessor fertiggestellt (Version 32/V), diese unterstützte jedoch keine virtuelle Speicherverwaltung.

Berkeley portierte 2BSD auf die VAX-11/780 und implementierte eine virtuelle Speicherverwaltung. Für VAX-Maschinen erschienen 1979 3BSD und dann 1980 4BSD. Die Entwicklung von BSD-Versionen für PDP-11s wurde unabhängig davon mit 2BSD weitergeführt, da der Adressraum der PDPs von nur 64 KByte zu klein für die neuen Versionen war.

1980 wurde ein Vertrag mit der Forschungsabteilung des amerikanischen Verteidigungsministeriums DARPA geschlossen. 3BSD und 4BSD sollten die Grundlage für die DARPA Entwicklungen, insbesondere im Bereich der Vernetzung werden. Dies führte zur Integration der Internet-Netzprotokolle (TCP/IP) in die BSDs.

Gleichzeitig beschloss AT&T ihre 7er-Edition (V7) zum System-III und dann zum System V weiterzuentwickeln und kommerziell zu vermarkten. Die neue Berkeley Version hieß daher nicht 5BSD, sondern 4.1BSD, um Verwechslungen auszuschließen. Mit 4.1BSD und den internen Versionen 4.1a, 4.1b und 4.1c wurde die Performance des Systems verbessert, erste TCP/IP Protokolle eingefügt und Netzwerkwerkzeuge (rsh, rcp) entwickelt. AT&T übernahm diese Entwicklungen später in ihr System V Produkt. Da die Berkeley Distribution immer noch frühen Quelltext von AT&T enthielt, verlangte AT&T den Erwerb einer (teuren) Lizenz von jedem, der Code aus Berkeley verwendete.

Aufstieg
Computerhersteller übernahmen die Berkeley Software Distribution und passten sie auf ihre Maschinen an. 1982 wechselte Bill Joy zur neu gegründeten Firma Sun Microsystems, die im selben Jahr die erste Version ihres auf BSD basierenden Betriebssystems SunOS herausbrachte. SunOS wurde im Lauf seiner Entwicklungsgeschichte um viele Funktionen des ohnehin lizenzierten System V erweitert, blieb aber lange Zeit seinen BSD-Wurzeln treu.

1983 brachte Digital für PDP-11 Ultrix-11 und dann für VAX Computer Ultrix-32 heraus. Ultrix basierte ebenfalls auf BSD. Es wurde später auch auf MIPS-Prozessoren portiert, die Digital in seiner Workstation-Linie verwendete.

Auch andere Unix-Hersteller verwendeten Teile der BSD. So setzte NeXT ab 1988 für ihr Betriebssystem NeXTSTEP ein BSD-System, allerdings mit einem Mach-Mikrokernel, ein. System V übernahm nach und nach Code von BSD, es bildete die Grundlage dieser Unix-Systeme.

Fall
Digital brachte um 1990 ihren Alpha Mikroprozessor heraus, der die VAX- und MIPS-Produkte ablösen sollte. Gleichzeitig wurde die Weiterentwicklung von Ultrix eingestellt, da auf Alpha das neue OSF/1 - ein Unix-Klon - angeboten wurde.

Sun bildete 1988 mit UNIX International eine Allianz mit AT&T und portierte ihr SunOS ebenfalls auf System V mit BSD Erweiterungen. 1992 erschien SunOS5 (auch Solaris2), das kein direkter Abkömmling der BSD-Distribution mehr war.

Damit war der letzte große Unix-Hersteller auf System V umgeschwenkt, alle Unix-Systeme verwendeten jedoch immer noch große Teile der BSD, darunter die von BSD eingeführten Kommandos und die TCP/IP-Netzwerkimplementierung.

Weiterentwicklungen bei Berkeley
Berkeley setzte seine Tradition fort und entwickelte die Distribution weiter. 1983 erschien 4.2BSD und 1986 4.3BSD. Es wurde klar, dass die VAX-Prozessoren durch andere Systeme abgelöst werden mussten, mit 4.3BSD-Tahoe (1988) wurde der Kernel in maschinenabhängige und portable Teile getrennt, 1990 erschien mit 4.3BSD-Reno eine Version, die unter anderem den Mach Mikrokernel unterstützte.

4.3BSD wurde sogar auf die PDP-11 Maschinen (zurück-)portiert und 1992 als 2.11BSD veröffentlicht. Der 250kByte große Kernel wurde dabei mit Overlay-Techniken auf den nur 64KByte großen Adressraum der PDP abgebildet.

BSD und AT&Ts Unix-Quellcode
Ende der 1980er hatte Berkeley so viele Erweiterungen eingebaut, dass nahezu der gesamte Unix-Quellcode von AT&T durch eigene Versionen ersetzt war. Hersteller von auf BSD basierenden Unix-Versionen mussten trotzdem wegen der übrigen Teile eine (teure) System V Lizenz von AT&T erwerben. Da unter anderem die gesamte Netz-

werkimplementierung von Berkeley stammte, hatten auch andere Hersteller Interesse an den Berkeley Entwicklungen, jedoch ohne eine AT&T-Lizenz erwerben zu müssen. 1989 gab die Universität deshalb die Networking Release/1 heraus, die alle von Berkeley identifizierten Dateien ohne Code von AT&T umfasste. Diese Version stellte jedoch kein vollständiges Betriebssystem mehr dar.

1991 erschien dann die Networking Release/2. Bill Jolitz ergänzte 1992 diese Release um nur sechs Dateien und veröffentlichte einen Patch, so dass ein vollständiges, fortgeschrittenes Betriebssystem für Intel 80386 Prozessoren namens 386BSD entstand.

Ebenfalls 1992 begann eine von der Universität Berkeley gegründete Firma namens BSDi (Berkeley Software Design Inc.) mit der Vermarktung der auch zum Betriebssystem erweiterten Networking Release/2. Sie vermarkteten ihr System unter dem Namen UNIX inklusive Quellcode für den Dumping-Preis von 995 Dollar. Die UNIX System Laboratories (USL), ein Zweig von AT&T, verklagten BSDI und die Universität daraufhin auf Einstellung des Verkaufs wegen Markenverletzung und teilweiser Verwendung ihres Quellcodes. Eine einstweilige Verfügung wurde jedoch abgelehnt.

Im Laufe dieses Rechtsstreits stellte sich heraus, dass AT&T Quellcode von Berkeley übernommen hatte (was wegen der BSD-Lizenz legal war), dabei jedoch die Urheberschaft Berkeleys aus dem Quellcode und der Dokumentation entfernt hatte (was in der BSD-Lizenz untersagt ist). Der Rechtsstreit endete 1994 damit, dass AT&T in einigen seiner Dateien die Urheberschaft von Berkeley wieder eintragen musste. Berkeley musste nur drei der über 18000 Dateien der Networking Release/2 entfernen und einige kleine Änderungen durchführen. Die Release war damit frei von Unix-Quellcode.

Im selben Jahr gab Berkeley dann die Version 4.4BSDLite als Nachfolger der Networking Release/2 heraus. 1995 war dann die Version 4.4BSDLite2 die letzte Version der Berkeley Software Distribution. 4.4BSDLite und Lite2 wurden zusammen mit 386BSD zur Grundlage von NetBSD, FreeBSD und kurz darauf OpenBSD.

Die BSDi entwickelte und vertrieb ihr System weiter unter dem Namen BSD/OS. Im Jahre 2001 wurde BSDi von Wind River Systems, Kalifornien aufgekauft.

Die Projekte NetBSD, FreeBSD & OpenBSD

Das freie 386BSD (1992) von Bill Jolitz zog Entwickler an, zumal PCs mit 80386 Prozessor sehr preiswert wurden. Jolitz war fest angestellt und hatte nicht immer genügend Zeit für die Fehlerbehebung und Einarbeitung von Verbesserungsvorschlägen. Dies veranlasste einige Entwickler im Jahre 1993 zwei Nachfolgeprojekte, NetBSD und FreeBSD, fast gleichzeitig ins Leben zu rufen.

Nachdem 1994 der Streit mit AT&T beigelegt war, brachten beide Projekte neue Versionen auf Grundlage von 4.4BSDLite heraus, das keinen AT&T Quellcode mehr enthält: NetBSD 1.0 (1994)

und FreeBSD 2.0 (Anfang 1995). 1995 zerstritt sich einer der Gründer des NetBSD Projektes, Theo de Raadt, mit den anderen Entwicklern und spaltete ein eigenes Projekt namens OpenBSD ab. Seit 2003 gibt es mit DragonFlyBSD, einer Abspaltung von FreeBSD, ein weiteres BSD Projekt. Auch das Betriebssystem Mac OS X und dessen Open Source Pendant Darwin von Apple, die nach dem Aufkauf von NeXT entwickelt wurden, basieren weitgehend auf BSD, speziell FreeBSD. Durch die Fortführung und Weiterentwicklung von NeXTSTEP als Mac OS X erfuhr BSD einen technologischen Aufschwung und eine sehr starke Verbreitung, die es mittlerweile als ähnlich wichtig wie Linux erscheinen lässt.

Gemeinsamkeiten & Unterschiede der BSDs

Die drei BSD-Projekte verstehen sich nicht als Konkurrenz, sondern als Ergänzung. Neuentwicklungen und insbesondere die zahlreichen Gerätetreiber eines Projekts werden von den anderen übernommen. Alle Projekte haben die Entwicklung eines freien Systems zum Ziel, ins eigentliche System soll daher nur mit der BSD-Lizenz kompatibler Code einfließen. Die Details der Interpretation des Begriffs freie Software unterscheiden sich jedoch geringfügig.

NetBSD möchte eine möglichst große Zahl verschiedener Hardwarearchitekturen und Prozessoren unterstützen. Of course it runs NetBSD. ist der Slogan des Projekts. NetBSD war lange nur über Downloads im Internet verfügbar und wendet sich vorwiegend an Anwender mit Unix-Kenntnissen.

FreeBSD konzentriert sich auf PC Hardware mit Intels 80386-Prozessor. Möglichst alle der erhältlichen Grafikkarten, Netzwerkkarten, etc. sollen erkannt und unterstützt werden. Eine einfache Installation ist das Ziel des Projekts, so dass auch Nichtfachleuten die Verwendung dieses Systems ermöglicht wird. Schon früh veröffentlicht das Projekt daher Installation-CDs. Neben Intel Prozessor werden heute auch andere Prozessoren wie Sun SPARC und Alpha unterstützt.

OpenBSD verfolgt dagegen das Ziel eines sicheren Systems. Dazu gehört zum einen proaktive Sicherheit, d. h. über Code-Audits sollen soweit wie möglich Fehler entdeckt und im voraus eliminiert werden. OpenBSD ist führend in der Implementation von sicheren, verschlüsselten Übertragungsverfahren. IPSec-Unterstützung gehört ebenso dazu wie eine offene Implementierung der Secure Shell ssh (OpenSSH).

Weitere BSD-Varianten: ArchBSD, ClosedBSD, Eclipse/BSD, EkkoBSD, emBSD, FireFlyBSD, FreeSBIE, GoBSD, GuLIC-BSD, HpBSD, MicroBSD, MirBSD, MirOS, NetBoz, PicoBSD, SecureBSD, theWall, TrustedBSD

Mac OS X

Mac OS X, welches momentan die Versionsnummer 10.3.8 (Panther) trägt, ist Appels aktuellstes Betriebsystem. Apple entwickelte dieses Betriebsystem um einen zukunftsorientierten Schritt in nächstes Jahrzehnt zu machen. Dieses Betriebsystem kombiniert neue, auf offene Standards basierende Technologien, die es zu einem kompatiblen, für eine breite Zielgruppe sehr interessanten Betriebsystem macht.

Architektur

Die moderne Architektur besteht aus sechs Hauptkomponeten: Die Systemprogramme, die Aqua-Benutzeroberfläche, diverse Programm-Frameworks, Standard basierende Grafiksysteme, umfassende Dienste für die Systemverwaltung und die Darwin Grundlage. Letztere soll auch im Vordergrund dieser Ausarbeitung stehen und wir werden uns nach einem kurzen Anschneiden der anderen Bestandteile von Mac OS X intensiv damit beschäftigen.

Die Systemprogramme

Für Mac OS X gibt es eine vielzahl von Systemprogrammen. Schon allein die Programme die sich nach einer Systeminstallation auf dem Rechner befinden, sind sehr umfangreich, funktionell orientiert und erhöhen den Spaß beim Arbeiten mit Mac OS X. Da das Haupteinsatzgebietes des „Macs“ nach wie vor die grafische Industrie ist, finden sich dort natürlich eine vielzahl der verschiedensten Programme zur effektiven Produktion mit dem Rechner.

Die Benutzeroberfläche

Apple versuchte bei der Aqua Benutzeroberfläche eine Fortführung der bisherigen OS 9 Oberfläche mit neuen, innovativen und zu gleich sehr komfortablen Funktionen zu erschaffen. Die Auqa Benutzeroberfläche ist noch flexibler, anpassbarer um dem Benutzer die bestmögliche Arbeitsumgebung zu bieten.

Die Programm-Frameworks

Max OS X enthält eine vielzahl umfassender Programm-Frameworks, welche auf traditionellen UNIX APIs (Application Programming Interface) basieren.
Aqua bietet mit seiner X-Code Entwicklungsoberfläche Entwicklern die Möglichkeit Cocoa (Diese besteht aus objektorientierten Frameworks durch die vorhandene Unix-Programme hinzugefügt oder völlig neue erstellt werden können.), Carbon, Apple-Script (Dient zur Automatisierung von Programmbefehlen und sich wiederholenden Aufgaben) und Java-Programme zu erstellen.
Hier muss unbedingt auf Carbon genauer ein

gegangen werden. Carbon ist eine Untergruppe der ursprünglichen Mac OS Toolbox APIs. Diese ermöglicht die Übertragung älterer Mac OS Software auf das neuen Betriebsystem.

Grafiken und Medien

Mac OS X bietet eine neuartiges Composing von Quartz für 2D, Open GL für 3D und Quicktime für Audio- und Videomaterial. Quartz ist eine hochleistungs Renderingbibliothek, welche auf dem Plattformübergreifenden PDF-Standart basiert und ist somit einzigartig. Mit Open GL hat sich Apple den Industriestandart für Visualisierung von 3D Objekten ins Boot geholt und QuickTime als Audio- und Videokomponete ist heutzutage Jedem ein Begriff. Zu erwähnen ist noch der Quartz Compositor. Dieser ist der Fenstermanager von Mac OS X, welcher auch sogenannte Low-Level-Dienste wie z.B. Cursor-Management bietet. Der Compositor basiert auf einem Vodeo-Mixer-Modell, welches jedes Pixel auf dem Bildschirm von allen Fenster in Echtzeit teilen kann. Dieses ermöglicht die eleganten Übergange der verschiedenen Zustände der Aqua-Oberfläche. Ab 10.2 (Jaguar) hat Apple Quartz Extreme auf den Markt gebracht. Dieses nimmt noch mehr „grafische"-Arbeit dem CPU ab und gibt diese den Grafikchips, ermöglicht gleichzeitig allen Programmen Zugang zu den 3-D Funktionen.

Systemverwaltung

Für die Systemverwaltung und den Netzwerkbetrieb vertraut Apple auf moderne Technologien um den heutigen Standart und Betriebsystemmix zwischen Windows, Mac OS, Linux und UNIX zu bestehen. Mehr möchten wir an dieser Stelle noch nicht darüber verlieren, da wir uns solchen Kernfunktionen des Betriebsystems später umfangreich wittmen werden.

OS Grundlagen

Was der normale Benutzer eigentlich gar nicht merkt ist, dass hinter der grafischen Oberfläche von Mac OS X die unix-basierende Open-Source Grundlage Darwin steckt. Zitat von Apple:

„Darwin stellt mit X11 und Posix Diesnsten eine komplette UNIX Umgebung bereit, die vergleichbar mit Linux ist."

Wie vergleichbar Darwin mit Linux ist und was noch so alles dahinter steckt wird Kernthema dieses Schriftstückes sein.

Geschichtliches zu Mac OS X

Um auf die Geschichte von Mac OS X einzugehen muss man gar nicht so weit zurückblicken, da Mac OS X nicht wirklich auf seinen OS Vorgängern basiert sondern eine komplette Neuentwicklung von Apple ist.

Da die klassischen OS-Versionen von Apple nicht der Ursprung von OS X waren werden wir deshalb hier nicht viel Zeit verlieren. Nur soviel Apple stellte fest dass im Konkurrenzkampf gegen andere Betriebsysteme und zum Bestehen in Zukunft eine radikale Überarbeitung des Betriebsystemes kommen musste.

Copland

Das 1993 entwickelte Betriebsystem sollte alle Anforderungen an ein stabiles, portables und leicht benutzbares Betriebsystem gerecht werden. Das

Betriebysytem, das als erstes auch multitasking-fähig sein sollte, scheiterte zum Ärger von Apple an einer Kombination aus Budegt- und Zeitüberschreitungen. Drei Jahre später wurde die Entwicklung des Betriebsystemes eingestellt.
Apple war nun auf der Suche nach einem neuen Fundament für Ihr zukunftsweisendes Betriebsystem.
Der nächste Schritt waren Verhandlungen mit „Be", die Firma die BeOS, ein fortschritliches Betriebsystem das Multitasking unterstützte und eine fortschritliches Dateisystem besitzte, entwickelte. Die Verhandlungen scheiterten ein paar Monate später am Preis, den Apple nicht zahlen wollte. BeOS selbst hielt sich auch nicht mehr lange über Wasser und wurde dann von Palm Inc. aufgekauft.
Nun war Apple mit einem Fundament für ein neues Betriebssystem immer noch nicht weiter. Aber es begannen die Verhandlungen mit NeXT Computer, welche zum dem Kauf des funktionsreichen System NEXTSTEP führten.

NEXTSTEP

NEXTSTEP wurde 1987 erstmals vorgestellt. NEXTSTEP hatte die Funktionen die Apple für ein zukunftsweisendes Betriebsystem benötigte. Es baute auf BSD-Unix und dem Mach-Kernel auf, es unterstützte Multitasking und Speicherschutz, zur Ausgabe auf dem Bildschirm nutze es Adobes Display-Post-Script, was den Entwicklern ermöglichte auf Drucker und Bildschirm das selbe auszugeben – echtes WYSIWYG. NEXTSTEP hatte auch schon eine GUI.
Der große Vorteil den NEXTSTEP hatte war dass es auf der Idee von Macintosh aufbaute ohne dessen ersten Fehler zu besitzen und wurde somit zur ersten Wahl für Regierung, Banken, Telekomunikationsbranche und im Gesundheitswesen.
NEXTSTEP wurde auf Sun SPARC, Digital Alpha und andere Systeme portiert. Später dann auch noch für Windows und Solaris angepasst, diese Version wurde dann unter dem Namen OpenStep bekannt. Folgende Dinge gingen von NEXTSTEP direkt in Mac OS X über: der Mach-Kernel, Das BSD-Layer und die NEXTSTEP-Frameworks, Bestandteile auf die später noch eingegangen wird.

Rhapsody

1997 stellte Apple Rhapsody vor, eine NEXTSTEP-Version, die auf Macintsoh-Hardware lief. Für 1998 kündigte Apple eine Mac OS Version für Power PC und Intel-Plattformen an. Diese sollte folgende Hauptkomponenten beinhalten: die Yellow-Box zur Ausführung von Open-Step Programmen, die BlueBox, die als zukünftige Mac OS Version bestehen sollte und die Red-Box welche es ermöglichen sollte Open Step-Applicationen auf Windows systemen laufen zu lassen. Rhapsody beinhaltete folgende Technologiene, welche heute Verwendung unter OS X finden:

Classic

Classic ist ein älteres Betriebsystem, welchen unter dem Neuen läuft, dies allerdings die neuen Funktionen wie Multitasking nicht unterstützt. Deshalb ist dies auch nicht die perfekte Lösung, aber eine Möglichkeit ältere Macintosh-Programme unter dem neuen System laufen zu lassen.

Java

Von Anfang an legte Apple viel Wert darauf, dass Java implementiert wurde, auf diese Art und Weise können Cocoa-Applikationen in Java Programmiert werden.
Zwei wesentliche Probleme waren dafür verantwortlich, dass Rhapsody nie auf dem Mark erschien:
1. Apple war es nicht möglich, das neuen System auf alle Apple Produkte zu portieren.
2. Die großen Software-Hersteller wie Adobe und Microsoft, waren nicht bereit ihre Software komplett neu zu programmieren, dass sie auf dem neuen System laufen würden, dies würde bedeuten dass alle Benutzer ihr Software auf der Classic-Umgebung laufen zu lassen. Apple wollte sich auch nicht ohne Unterstützung der Softwarehersteller weiterbewegen.
Aus diesen Gründen hat sich Apple die Sache mit dem neuen System noch einmal überlegt und kam zu dem Entschluss sich die Zeit zu nehmen das System für eine weitere Palette von Hardware zu portieren und Kompatibilitässchicht, für ältere Mac-Software zu entwickeln diese wurde 1998 unter dem Namen Carbon vorgestellt.
Nun möchte man meinen, Rhapsody war, da es keine Vermarktung erfuhr, ein weiterer Fehlschlag. Dies ist aber falsch, ohne Rhapsody gäbe es heut kein Mac OS X oder etwas ähnliches. Es war so, dass in Rhapsody wurde das eben genannte Carbon eingebunden wurde und es wurde in Mac OS X getauft.

„Endlich ein neues Betriebsystem"

Wie schon erwähnt ist Darwin (von Apple so getauft) der Betriebsystemkern von Mac OS X. Der auf BSD UNIX und Mach 3.0 Mikrokernel basierende Kernel ist seit seinem Release 1999 bei Apple im Einsatz. Apple hat ja 1997 sein KnowHow von NeXT übernommen, welche zuvor Nextstep und danach das Betriebsystem Openstep entwickelt haben, welches auf BSD 4.4 basiert. Als Refernez für Mac OS X hat Apple die Distribution Free BSD herangezogen, was Apple ermöglicht OpenSource Projekten kompatibel zu sein. Nicht zu vergessen ist aber dass Net BSD und Open BSD auch bei der Entwicklung zu Darwin eine entscheidende Rolle spielten. Die integreirte Kryptographie und Open ssh stammen aus Open BSD und viele Werkzeuge und Kommandos aus Net BSD. Der Darwin-Kernel nennt sich XNU und besteht aus Mach und BSD. Mach stellt das Speichermanagement, IPC und Messaging, I/O-Bereiche (Gerätetreiber) zur Verfügung. BSD verwaltet die Benutzer-, Zugriffsrechte, den Netzwerkstack, das virtuelle Dateisystem (VFS) und die POSIX Kompatibilitätsschicht.
Ein weiterer Vorteil, den Apple dank BSD zu nutzen macht ist, das sobald die OpenSource-Gemeinde eine Sicherheitslücke bei BSD ausgemerzt hat, tut dies Apple auch - wer außer Apple kann sagen „meine Entwickler sind weltweit in den Kellern der Informatiker verteilt"
Unter dem OpenSource Konzept arbeitete Apple an dem für Power PC und X86 Systemen kompatiblen Version. Und wie es sich für ein richtiges Open Source Projekt gehört fließen alle Weiterentwicklungen wieder in den öffentlichen Pool zurück.

Grundlagen

Der Mach 3.0 Microkernel

Eine weitere Grundlage, die wir zuerst legen wollen, ist der Mach 3.0 Microkernel.
Ein Microkernel ist der eiegntliche Systemkern eines Computers und ist für die Kommunikatioinskanäle zwischen den User-Mode-Modulen für die funktionellen Aufgaben zuständig. Der Kernel wird meist in den Sprachen C oder C++ programmiert. Der Kernel an sich besteht nur aus den wichtigsten Funktionen der Prozessverwaltung.
Die Carnegie Mellon University entwickelte zwischen 1985 und 1994 einen Microkernel, der bis heute noch Verwendung in Betriebsystemen findet. Dieses Projekt nennt sich Mach. In den meisten Fällen, in denen der Kernel verwendet wird weicht dieser vom typischen Mach-Design ab. Auch NEXTSTEP hat verschiedene Erweiterungen für diesen Kernel und so kommt es dazu, dass es schon die Version 3.0 des Mach-Kernels gibt.

Shell-Script

Um nicht später ein störendes Stocken hervorzurufen, werden wir noch den Begriff „Shell-Skript" erklären.
Ein Shell-Script ist eine Textdatei in der Komandos gespeichert sind. Diese Textdateien sind in einem vom Menschen lesbaren Code geschrieben und werden erst während des Ausführens interpretiert, dies hat den Vorsteil, dass der Benutzer ein solches Script erweitern, bzw auf Fehler untersuchen kann. Shell-Scripte dienen der Arbeitserleichterung, da sie mehrere Kommandos beinhalten und somit zusammenfassen.

swap-Partition

Dies ist ein Speicherbereich der festgelegt wurde zur Datei-Auslagerung für den Fall, dass der Arbeitsspeicher nicht mehr ausreichen würde. Auf den swap-Breiecht hat nur das Betriebsystem Zugriff.

Mac OS X Bootvorgang

Nach dem wir versucht haben die Grundlagen zu geben, werden wir uns nun wieder dem Mac OS X wittmen und was ist nicht sinnvoller als mit dem Systemstart zu beginnen.
Im Normalfall drückt man auf den Startknopf des Macs warten wenige Augenblicke und beginnt zu arbeiten ohne, dass man sich einen Kopf darüber macht, was der Rechner während des Starfvorganges so alles macht.
Nachfolgend zeigen und beschreiben wir die einzelnen Schritte des Bootvorgangs in der Reihenfolge wie sie abgearbeitet werden.

1. Das BootROM

Das BootROM ist wie jeder andere ROM ein Hardwarebaustein auf dem Motherboard. Es enthält den Code für den POST und die Open Firmware, der ausgeführt wird, sobald der besagte Einschaltknopf auf dem Computer gedrückt wird.

POST

Power On Self Test. Dieser wird bei jedem Systemstart ausgeführt – nachdem der Computer vollständig heruntergefahren war. D.h. der POST wird nicht bei einem Neustart ausgeführt.
POST initiallisiert die wichtigsten Hardwareschnittstellen (RAM, PCI, SCSI, FireWire und USB) und

stellt sicher, dass genügend funktionierendes RAM vorhanden ist, ein defekter RAM-Baustein wird vom POST deaktiviert.
Wenn alles in Ordung ist wird der Start eingeleitet, wenn nicht gibt es diverse Rückmeldungen in Form von Systemklängen, auf die wir jetzt nicht eingehen wollen.

Open Firmware

Die OpenFirmware ist eine rudimentäre Programmumgebeung, die neben dem POST auf dem BootROM gespeichert ist. in der OpenFirmware wird ein in der Sprache „Forth" geschriebenes Programm ausgeführt, dass die Informationen aller angeschlossenen und einegschaltenen Geräte abruft und daraus einen Device-Tree erstellt.
Mit gewissen Tastenkombinationen, wie z.B. „C" für starte von CD-Rom, kann man den Startvorgang mit Hilfe der OpenFirmanware modifizieren.
Die OpenFirmware ist auch verantwortlcih für den NVRAM (auch bekannt als Parameter-RAM). Dies ist ein nichtflüchtiger Teil des Arbeitsspeichers, der gewisse Informationen für den Start und Betrieb des Rechners speichert.
Im NVRam ist auch der Pfad der bootinfo-datei gespeichert, von der gestartet werden soll. Für diese Datei gibt es einen Standartwert und einen Beutzerwert, der den Wünschen des Benutzers angepasst werden kann. Um den NVRAM einzusehen können Sie im Terminal folgenden Befehl eingeben

```
$ nvram -p
```

Die Ausgabe wird in etwa so aussehen:

Ist eine Startfähige bootinfo-datei gefunden (z.B. BootX), wird diese in den geschütztden Arbeitsspeicherbereich geladen und gestartet. Nun kann die OpenFirmware, die Kontrolle übergeben und beendet werden. Mehr Informationen zur Open Firmware geben wir im Kapitel „Alltagstipps".

2. BootX

Wenn die OpenFirmware Mac OS X als zu startende Betriebsystem auswählt, übernimmt BootX die vorläufige Kontrolle über den weiteren Startvorgang. Die Ausgabe von BootX ist das Laden der Kernelumgebung, damit später die Systemkontrolle dem Kernel übergeben werden kann. Wenn man sich mal einen Mac OS X Systemstart vorstellt, sind wir jetzt an der Stelle, an der man das sogenannte „booting Image" ein grauer Apfel angezeigt wird. BootX versucht als erstes, die Gerätetreiber aus dem „kextcache" zu laden. Diese Treiber wurden bei einem früheren Systemstart vorbereitet und komprimiert im Bytecode zwischengespeichert, so dass sie schneller geladen werden. Es handelt sich um Gerätetreiber, welche unabdingbar sind, wie z.B der der Grafikarte, oder der Festplatten.

Ist dieser Cache defekt, oder gar nicht vorhanden, z.B. bei ersten Systemstart, sucht BootX im Verzeichnis „system/library/extensions/" nach Treibern mit dem Eintarg „OSBundleRequired". dieser Eintrag kann folgende werte haben:

Root
Diese Treiber werden immer geladen.

Local-Root
Diese Treiber werden bei einem normalen Start von einem lokalen Volume geladen

Network-Root
Diese Treiber werden bei einem normalen Netzwerkstart geladen

Console
Diese Treiber werden bei einem Start in den Single-User Modus zusätzlich zu den Root-Treibern geladen

Safe Boot
Diese Treiber werden bei einem Safe Boot zusätzlich zu den Root-Treibern geladen

ohne Eintrag
Diese Treiber werden später von kextd dynamisch hinzugeladen, wenn sie gebraucht werden.

Wenn alle nötigen Treiber geladen sind, wird der Kernel initialisiert – d.h. es werden die nötigen Verbindungen geschaffen, so dass der Kernel über die Treiber mit der Hardware richtig kommunizieren kann.

3. Systeminitialisierung durch den Kernel
hat BootX seine Arbeit getan, kann der Kernel die Kontrolle über den weiteren Start übernehmen. Er startet den Prozess „mach_init", welcher seinerseits den Prozess „init" startet. „init" trägt immer die Prozess-ID 1 und ist quasi der Eigentümer aller anderen Systemprzessse.
„init" prüft unter anderem ob eine der folgenden Tastenkombinationen angewählt wurde.

1. Apfel + S
Starte im Singel-User-Modus, „init" führt keine weiteren Aufgaben aus, sondern übergibt die Kontrolle dem Benutzer.

2. Apfel + V
Starte im Verbose-Mode (verbose = wortreich), d.h. das „init" den weiteren Bootvorgang nicht grafisch darstellt sondern die Ergebnisse in einer Textkonsole auf dem Bildschirm zeigt.

Wenn nicht im Single-User-Modus gebootet wurde ist „init" noch für folgende Prozesse zuständig:

- für das Starten der Prozesse, die für die Login-Przedur nötig sind.
- für das Starten der beiden Prozesse „/etc/rc.boot" und „/etc/rc" welche nun beschrieben werden.

Weitere Startoptionen
bevor wir mit dem systemkstart weitermahcen wollen wir noch andere Startoptionen erklären
Die folgenden Tasten müssen während des Starforganges gehalten werden

C
Start von CD-Rom

D
Start von der internen Festplatte

N
Netzwerkstart

R
Zurücksetzen des Displays bei Powerbooks und ibooks

Apfel, Alt, Umschalt, Entfernen
ignorieren des Start-Volumes und suchen nach einem anderen starfähigen Volume.

T
Den Rechner in Fire-Wire-Target-Modus versetzen, dann kann es als Festplatte genutzt werden.

Alt
Ausführeder Anwendnug zur Auswahl des Startvolumes

X
Starten im Safe-Modus

Apfel, Alt, O, F
Direkter Zugriff auf die Open Firmware

Apfel, Alt, P, R
Zurücksetzen des NV-Rams (Parameter-Ram)

Maustaste gedrückt halten
Auswerfen eines Mediums, z.B. einer CD-Rom

Aber nun wieder zrück zum Systemstart

rc.boot
der „init"-Prozess startet nach seiner Initialisierung des Kernels das Shell-Script „/etc/rc.boot". Das Script führt abhängig je nach Art des Bootvorgangs eines entsprechenden Check des Filesystems (fsck) durch und gibt die entsprechneden Hinweise an die Konsole aus:

netboot
kein Filecheck, Übergabe der Kontrolle an „/etc/rc.netboot"

lokale HD / SingleUser
Es wird kein fsck, sondern ein Hinweis mit entsprechendem Befehl an die Konsole ausgeben.
lokale HD / Safeboot
Es wird ein fsck mit den Optionen -f und -y durchgeführt. das heisst, auch Filesysteme mit aktiviertem Journaling werden überprüft.

CD-Rom
kein Check, da auf einer CD-Rom ohnehin nicht geschrieben werden kann.
Wenn der Filecheck keine Fehler findet, wird das Script beendet, andernfalls wird ein erneuter Check durchgeführt und das System wird neu gestartet.
In seltenen Fällen kann es vorkommen dass ein Dateisystem wegen eines Hardwaredefetes, .z.B. defekte Festplatte, unrepaierbar ist. Der Computer käme bei einem Start nie über das rc.boot-Script

hinaus, weil dieses immer wieder ein Fehler entdecken würde. Man sollte dann versuchen im Single-user-Modus zu starten und von dort aus seine Daten zu retten.

rc

Nachdem das rc.boot ohne Fehler erfolgreich ausgeführt wurde, wird das Shell-Script „/etc/rc" von „init" gestartet. das „rc"-Script startet zuerst das Subscript rc.common, welches die Standarteinstellungen für die Shell vornimmt und einige Funktionen und Konstanten definiert. Anschließend erfüllt das Script folgende Aufgaben.

1.

ermitteln der Startoptionen „Verbose-Modus" oder „Single-User-Modus" – abspeichern der Resutate zwecks der Wiederverwendung.

2.

Ermitteln der BootType: automatischer Neustart (autoboot), oder normaler Neustart (multiuser).

3.

falls von einer CD-Rom gestartet wird, Übergabe der Kontrolle an „/etc/rc.cdrom"

4.

ermitteln ob ein Netbootdurchgeführt wird – abspeichern der Resutate zwecks der Wiederverwendung.

5.

Mounten der lokalen Volumes

6.

Start des Kernel-Extension-Dämons „ketxd" der auf Anfrage des Kernels zusätzliche Extensions läd, bei einem Safeboot werden nur die Standart-Extensions geladen.

7.

Wenn nicht vorhanden wird das mkext-cache erstellt, so dass spätere Systemstarts etwas schneller ablaufen

8.

Initialisieren des Arbeistspeichers und Anlegen einer ersten swap-Datei.

9.

Start des System-Starters mit den Optinonen -v (verbose) oder -x (safeboot), falls diese bei 1. gespeichert wurden.

10.

Beenden des Scriptes

4. SystemStarter

das Boot-Script „rc" ruft den SystemStarter auf. Der Systemstarter ist ein kleines Programm welches die Startupitmes startet. Das Programm liegt in „System/Library/Startupitems". Der SystemStarter kennt die verschidenen Optionen z.B. „-v" und weiß genau, ob er nun die grafische Oberfläche zeigt oder eine Konsole, welche den Fortschritt des Systemstartes ausgiebt. Sobald der Systemstarter genügend Diesnte gestartet hat, kann das Login-Fenster gezeigt werden.

5. LoginWindow

der Prozess „LoginWindow" wird von „init" gestartet und ist für das An-, Ab und „Um"-melden der Benutzer zuständig.

System-Profiler

Wenn Sie nun Ihr System gestartet haben, ist es wohl interessant zu erfahren, wie Ihr System ausgestattet ist, dazu bietet sich an den System-Profiler zu verwenden. Wir wollen hier den System-Profiler auf der Kommandozeile verwenden, den es natürlich auch auf der GUI gibt. Dazu starten wir das Terminal und geben folgendes Befehl ein

```
$ system_profiler
```

Dieser erstellt uns jetzt einen vollständigen Bericht über unser System. Da dieser Bericht sehr lang ist und es wohl übersichtlicher ist diesen in einer Textdatei zu haben, können Sie dies mit folgendem Befehl erledigen

```
$ system_profiler > ~/Desktop/mein-Systemprofil.txt
```

tun.

Es gibt nun auch ein paar Möglichkeiten die Ausgabe des System-Profilers zu Filtern dazu müssen Sie folgende Optionen an den Befehl anfügen

```
$system_profiler -listDataTypes
```

nun sehen Sie eine Auflistung der Datentypen Ihres Systems. Eine weitere Möglichkeit ist sich nur die Informationen der Netzwerkverbindungen anzeigen zu lassen

```
$system_profiler -SPNetworkDataType
```

Das Filesystem von Mac OS X

Jeder der schon einmal Datein von Mac OS X auf ein anderes System (z.B. Linux) ausgetauscht hat, dem ist bestimmt schon einmal aufgefallen, dass es zu jeder Datei einen zweiten File gibt. Um ein Beispiel zu nennen bei einem Bild Names „Haus.eps" gibt es einen File „._Haus.eps", aber warum ist das so ?

Um dies zu erklären muss man zeitlich etwas zurückkehren und zwar zu den Vorgängern von Mac OS X.

Wenn ein sogenanntes klassisches Mac OS, Dateien speichert, besteht jede Datei aus zwei Dateizweigen einmal dem Daten-Zweig und dem Resourcen-Zweig. Der Daten-Zweig enthält logischerweise die Daten des Files, also z.B. den ausführbaren Code eines Programmes. Der Resoursen-Zweig enthält natürlich die Resourrcen welche welche z.B die Angaben über die Optik von Programmmenüs und Programmicons.

Der Vorteil den sich Apple damit erarbeitet hatte ist, dass ein Programm aus nur einer Datei bestehen kann. was auch die beliebte Funktion von Classic-Systemen erklärt, ein Programm einfach von einem auf den anderen Rechner ziehen und es läuft!

Allerdings hat der wandel der Zeit gezeigt, dass es doch immer wieder Programme gibt, die auf Grund ihrer Komplexität doch mehr als eine Datei benötigen.

Ein weiteres Problem ist die Kompatibilität mit anderen Systemen, da diese die Macintosh-Resourcen-Zweige nicht lesen können. Daher müssen die Dateien z.B. in das Binhex-Format gewandelt werden, dabei werden nämlich alle in dem Re-

sourcenszweig gespeicherten Daten in eine Datei geschrieben und kann dann auch von anderen Systemen gelesen werden.
Dies ist allerdings gar nicht so schwer wie es klingt, Komprimiert man eine Datei auf einem Macintosh-recher (.hqx) hat man automatisch die Datei konvertiert.
Das Format, dass Mac OS benutzt hat um Dateien zu sichern nennt sich HFS (Hirarchical File System). Da dieses Format Probleme hat größere Volumes zu lesen, hat Apple HFS+ einegführt. Dieses System hat folgende Merkmale:

- Pfadnamen werden mit „::“ anstelle von „/“ geschrieben
- Es wir mit File-IDs gearbeitet, das heißt eine Datei wird nach Nummer zugeordnet nicht nach physikalischen Pfad was den Vorteil hat, dass man Ordner oder Volumes problemlos um bennen kann ohne das Dateiverknüpfuneg veroren gehen.
- HFS+ ist NICHT casesensitiv
- HFS+ kann sogennate Meta-Dateien speichern.

Diese Dateien werden von Unix-Systemen verwendet um Zugriffsrehte zu vergeben. Diese Dateien können aber erst von Mac OS X gelesen werden, das hat zur Folge, dass man eine Rechte beschrenkte Datei problemlos mit einem Classic-System öffen kann.
Ein weiterer Unterschied zu einem anderen System ist, dass die anderen Systeme immer ein Suffix an einer Datei benötigen um diese richtig zuorden zu können. Ein Mac OS Benutzer ist der Meinung dies nicht zu müssen, aber das ist nicht ganz richtig. Den Mac OS benötigt für die Zuordnung von Datein sogenannte Creator- und Type-Codes, die dem System sagen um welche Art von Datei es sich handet. Diese wollen wir noch genauer erklären.

Type- und Creator-Codes

Der Type-Code ist ein aus 4 Zeichen bestehender String, der einer Datei bei ihrer Erstellung zugewiesen wird und dem System mitteilt um welche Art von Datei es sich handelt. Zum Beispiel:

Suffix	TypeCode
.pdf	PDF
.doc	W8BN
.xls	XLS8

Der Creator-Code ist ähnlich wie der Type-Code nur dass er das Programm angiebt mit dem die Datei erzeugt wurde. Dieser ist auch wieder 4 Zeichen lang.

Creator Code	Programm
prvvw	Vorschau
MSWD	MS Word
XCEL	MS Excel

Um wieder auf Mac OS X zurückzukommen. Hier wollte man auf das System mit den zwei Dateizweigen verzichten, da man sich aber immernoch in HFS befindet, unterstützt Mac OS X die Rescorcendateien, versucht aber so oft wie nur möglioch darauf zu verzichten. Das bedeutet aber nicht, dass es unter OS X keine Resourcen

mehr gibt – nein, diese sind jetzt aber eine Datendatei mit die dann mit einem Punkt und Unterstrich „._Beispiel"beginnt. Und was hat sich Apple ausgedacht, dass man diese nicht sieht, Mac OS X zeigt einfach keine Dateien die mit einem „." beginnen - ob da so toll ist ?
Da man weiterhin dem Macintosh Benutzer die Möglichkeit bieten möchte Programme wie vor her beschrieben, durch darg & drop zu installieren hat Apple die sogennten Bundels eingeführt.

Bundels
Bundels sind für den Benutzer einzelne Datein die er auf dem Volume angezeigt bekommt. Allerdings verbergen sich dahinter eine Art Ordner die gleichzeitig wie ein Alias funktioniert. In diesem Bundel befinden sich die Datei und die Resourcen die für das jeweilige Programm benötigt werden.
Die Bundels werden mit einem Doppelklick ausgeführt mit der ctrl - Taste und einem Klick kann man über das Kontext-Menü „Den Packetinhalt zeiegn"

Journaling Files System
Seit Mac OS 10.2.2 gibt es von Apple ein sogenanntes Journaling File System (JFS). Das JFS ist eine optionale Extension von HFS+ die, wenn sie aktiviert, ist dafür da ist, dass der Kernel selbstständig alle Veränderungen am Dateisystem protokolliert. Das hat den Vorteil, dass ein defektes System schneller wiederherzustellen ist.
Ein Kernel schreibt während der Laufzeit des Rechers ein „Inhaltsverzeichnis". Bei einem Systemabsturz kann es nun passieren, dass das Inhaltsverzeichnis mit dem Datenbestand nicht mehr übereinstimmt, was immerhin bedeuten könnte, das das System defekt ist. Bei einem Neustart müsste ein Filecheck geamcht werden, was bei einem großen Server (daher kommt auch die Technik) einen emensem Zeitaufwand bedeuten würde. Wenn nun durch JFS ein „Journal" vorhanden ist, kann das Dateisystem einfach auf den letzten sicheren Stand gebarcht werden, was viel schneller von Statten geht.

Fragmentierung
Jedem Benutzer eines Computers ist der Fragmentierung bzw. Defragmentierung ein Begriff. Unter Fragmentierung versteht man das Problem, dass wenn Dateien über Ihren zugewiesenen Platz hienauswachsen, das System die Dateien in vielen unzusammenhängende Teile an mehreren Orten auf der Festplatte verteilt. Dies führt zu Performanceeinbusen, da das System nicht direkt auf die Daten zugreifen kann, sondern diese erst „zusammensuchen" muss. Dem wurde dann mit einer Defrtagmentierung der Festplatte entgegengewirkt.
Seit Panther hat Mac OS X seine ganz eigene Art mit diesem Problem umzugehen, Voraussetzung ist allerdings, dass JFS aktiviert ist. Panther hat drei Optimierungen, die wir hier aufzeigen möchten:

Automatic File Deframentation
Wenn eine Datei geöffnet ist,die mehr als acht Fragemnte hat und kleiner als 20 MB ist wird diese einfach an einen anderen Ort verschoben, an dem sie „am Stück" gespeichert werden kann.

Adaptiv Hot File Clustering
Wenn das system kleinere Dateien entdeckt, auf

die öfters zugegriffen wird und welche nicht gespeichert werden, verschiebt es diese an Stellen der Festplatte, die einenen schnelleren Zugriff erlauben, Dateien die dieses Kriterium nicht erfüllen werden vom System von solche Stellen ausgelagert.

Automatische Defragementierung
Das System führt nach einer installation eines Programmes automatisch eine Defragmentierung der Festpallte durch.

Weitere Dateisysteme
Mac OS X unterstützt zu HFS+ und wie schon erwäht HFS auch noch andere Dateisysteme die an dieser Stelle genannt werden sollten:

UNIX File System
Dies ist eine Variante des BSD Fast File System. Unix-Entwickler arbeiten genre mit UFS, allerdings bringt es unter Mac OS X nicht die Performance die man mit HFS+ erreichen kann.

Universal Disk Format
Dies ist das Standartformat für alle DVD-Medien und ermöglicht Mac OS X im Gegensatz zu seinen Vorgängern mit DVD+R Medien umzugehen

ISO 9660
Das ist der Standard für plattformübergreifende Formate für CD-Roms, die hat es aber auch schon bei den Classic-Systemen gegeben

FAT
Das Standartdateisystem von MS Dos „File Allocation Table“, dies wird bei den meisten Windowssystemen verwendet.

Netzwerkbasierende Dateisysteme
Die Unterstüzung von netzwerkbasierenden Dateisystemen ist in dem Moment wichtig, in dem Sie versuchen mit ihrem Rechner in einem Netzwerk ein anders Laufwerk zu mounten.

Apple Filing Protokoll (AFP)
das ist das macintosheigene Protokoll für FileSharing. Urspünglich wurde es als Apple Talk entworfen aber unterstützt jetzt auch IP-basierende Netzwerke. Dsa AFP wird hautsächlich zur Verbindung von Macintosh-Computern verwendet.

Service Massage Block (SMB)
Common Internet File System (CIFS)
Das Windows-eigene Netzwerkprotokoll. Dieses ist eigentlich als SMB bekannt. Microsoft hat mit CIFS einst versucht das Protokoll zu standardisieren, was nie gelungen ist. Dieses Protokoll wurde von der Software-Suite Samba in Mac OS X implementiert.

Network File System (NFS)
Dies ist ein Unix-Netzwerkdateisystem wie es von Linux benutzt wird.

Web-Based Distributed Authoring and Versioning (WebDav)
Das ist ein Protokoll mit dem Dateien auf einem Webserver gemeinsam mit anderen verwaltet und bearbeitet werden können.

File Transfer Protokoll
Ein Protokoll das zur Datenübertragung zweier Rechner verwendet wird. Dies findet hauptsachlich im Internet Verwendung.

Die Verzeichnisstruktur
Bei der Benutzung von Mac OS X mit der GUI, wird dem Benutzer von Auqa etwas vorgemacht. Den Aqua zeigt dem Benutzer nur die Ordner, die er benötigt um mit Mac OS X zu arbeiten. Diese sind für uns im Moment nicht so sehr interessant, deshalb erwähnen wir diese hier nur kurz.

Applikations (Mac OS 9)
Hier leigen die Programme für die Benutzung der Classic-Umgebeung

Benutzerhandbücher und Informationen
Der Ordnername verrät ja schon den Inhalt.

Library
Dieser Ordner enthält programm- und systemspezifische Ressourcen, wie Schriftarten und Programmvoreinstellungen, die von allen Benutzern des Systemes benötigt werden.
Programme
Hier liegen die Programme für Mac OS X

System
Dieser Ordner enthält den Unterordner Library, dieser enthält wiederrum essentielle Systemwerweiterungen.

Systemordner
Dieser wird für den Start der Classic-Umgebung benötigt.

Wie schon erwähnt gibt es auf dem Rechner noch andere Verzeichnisse die Aqua uns vorenthalten will, aber diesen möchten wir uns genauer wittem. Die Verzeichnisse können mit Hilfe des Treminals betrachtet werden. Die oberste Ebene, die wir jetzt auch betrachten wollen, nennt man die „root-Ebene“ .Man startet dazu das Terminal und gibt folgenden Befehl ein:

```
$ ls /
```

folgende Verzeichnisse sollten nun zu sehen sein (eventuell sehen Sie auch noch mehr Dateien, dies ist je nach Installation des Rechners verschieden):

Desktop DB
Desktop DF
Hinter diesen Dateien befindet sich der Mac OS9 Schreibtisch. Der Schreibtisch-Ordner im Finder benutzt diese Dateien, um Ihnen den Inhalt des Mac OS9 Schreibtisches zu zeigen während Sie sich in Mac OS X befinden.

/Network
Dieses virtuelle Dateisystem ermöglicht das browsen durch Dateisysteme von Servern, mit denen man verbunden ist.

/Volumes
Dieses Verzeichnis enthält neben dem Start-Volume alle anderen Medien mit denen Ihr Rechner verbunden ist.

/automount
Dieses Verzeichnis enthält alle Verknüpfungen zu Netzwerkservern, die mit Ihrem Mac durch eines der Protokolle AFP, NFS oder SMB verbunden ist.

/bin
das Verzeichnis bin enthält Binaries, das sind maschinenlesbare Programmcodes für verschiedene fundamentale Shellbefehle wie z.B „ls" und die verschiedenen Shells selbst.

/sbin
Dieses Verzeichnis enthält weitere Binaries. Hauptsächlich sind die System-Programme und Administrator-Utilities.

/cores
Dies ist der Ort an dem die UNIX Programme ihre Crash-Datein ausgeben.

/dev
Die Geräte, die unter UNIX gemountet werden werden immer in dieses Verzeichnis gemountet so kann das System hierüber die am Compunter angeschlossenen Geräte ansprechen.

/privat
In dem Verzeicnis privat liegen nun Unterverzeichnisse, die bei „normalen" UNIX Systemen auf der Root-Ebene liegen. Diese sind:

privat/etc
hier liegen einige log-Datein, config-Datein und Shell-Scripte

Privat/tmp
hier liegen temporäre Dateien,die bei einem Neustart verloren gehen.

privat/etc
hier liegt das Homeverzeichnis von „root" (varia), temporäre Dateien (tmp) die beim Neustart nicht verloren gehen und ein Verzeicnis (VM) worin der virtuelle Speicher seine swap-Files ablegen kann.

/usr
im Gegensatz zu den anderen Verzeichnissen, die Dateien und binaries des Systemes beinhalten, beinhaltet diese die Unterverzeichnisse des Systemadministratrors.

/usr/bin
allgemeine Utilities und Programme

/usr/lib
diverse Librarys

/usr/libexec
System-Utilities, die von anderen Programmen gestartet werden – Hintergrundprozesse

/usr/share
System-Daemons und Utilities, die vom Benutzer gestartet werden

/var
In diesem Verzeichnis befinden sich Dateien, welche sich oft ändern, wie Mail-Spools und Log-Dateien

Im Übrigen sehen wir hier auch den „mach_kernel" – wie wir ja wissen das Herz unseres Betriebsystemes. Die beiden Dateien „mach" und „mach.sym" gehören auch dazu.

Nun gibt es noch eine Liste weiterer versteckter Datein, welche mit dem Befehl **„`$ ls -a`"** gezeigt werden können. welche hier noch zu erklären wäre ist die:

.DS_Store
Diese Datei enthält die Finder-Einstellungen sowie die Kommentartexte zu den jeweiligen Dateinen in einem Ordner.

Wer der Meinung ist, dass er diese ganzen versteckten Dateien auf dem Finder sehen möchte, muss dazu folgende Zeile ins Terminal tippen:

```
$ defaults write com.apple.Finder
ApplesShowAllFiles YES
```

der Finder benötigt ein Moment um die neue Einstellung anzunehmen. Um diese Einstellung zu wiederufen gibt man einfach

```
$ defaults write com.apple.Finder
ApplesShowAllFiles NO
```

ein.

Zugriffsrechte

In diesem Kapitel wollen wir nicht erklären wie man Zugriffsrechte in der GUI verändert, da dies keine Schwierigkeit darstellen sollte. Hier soll erleutert werden wie man die Zugriffsrechte im Terminal verändert.
Dazu sollten aber zuerst die Zugriffsrechte erleutert werden. Dazu haben wir hier ein Beispiel, an dem wir die Zugriffsrechte erklären möchten:

drwx — —	14	daniel	users	476	25 Jan	22:10	Bilder
drwx — —	2	daniel	users	266	18 Jan	15:12	Dokumente
drwx r-x r-x	7	daniel	users	123	12 Jan	22:10	Musik
drwx r-x r-x	18	daniel	users	230	20 Feb	15:12	Schriften

Dateiart
Zugriffsrechte
Besitzer / Gruppe / Alle
Verknüpfungen
Benutzer
Gruppe
Dateigröße
Änderungsdatum
Name

Um eine solche Ansicht zu bekommen muss man den befehl **„`ls -l`“** im Terminal eingeben.
Die Buchstaben, die die Zugriffsrechte beschreiben haben folgende Bedeutungen:

r – Leserecht für Datei oder Verzeichnis
w – Schreibrecht für Datei oder Verzeichnis
x – Datei kann ausgeführt werden.

Zusammenfassung

— – Kein Zugriff
r– – „Nur-Lese“-Zugriff
–x – „Nur-Ausführ“-Recht
r-x – „Lese und Ausführ“-Zugriff
-wx – „Schreibe und Ausführ“-Zugriff
rwx – Vollzugriff

Zugrissrechte vergeben
Da wir ja nun wissen was Zugriffsrechte sind, sollten wir als nächstes erklären, wie man diese ändert bzw. vergibt.
Das tun wir mit dem Befehl „chomd“

chmod
Der Befehl chmod ist eine Abkürzung für „change File Modus“ - Zugriffsrechte einer Datei ändern. Der Befehl „chomd“ baut sich folgendermaßen auf:

```
chmod „zugriffsoption“ Datei
```

Die Zugriffsoptionen zeigen an was mit den Zugriffsrechten, für wen geschehen soll. Deshalb benötigt man auch einen Code für „wen“ (u, g, w, a) eine „Aktivierung“ (+, -, =) und ein Zugriffsrecht (r, w, x). Diese werden wir nun folgt erklären:

wer

u	Benutzer
g	Gruppe
o	Andere
a	Alle

Aktion

+	Zugriffsrecht ergänzen
-	Zugriffsrecht entfernen
=	Nur dieses Zugriffsrecht

Zugriffsrecht

r	Lesen
w	Schreiben
x	Ausführen
t	Sticky-Bite (wird später erklärt)

Zum besseren Verständnis wollen wir hier ein paar Beispiele für die Zugriffsrechtevergabe geben:

Um jeden Schreibrechte an Ihrer Datei zu geben

```
$ chmod a+w meinDokument.doc
```

Um jedem der Gruppe ein Ausführungsrecht zu geben

```
$ chmod g+x meinProgramm
```

Um jedem außer Ihnen selbst das Recht eine Datei zu lesen verwehren

```
$ chmod go-r privat.doc
```

Mit der Option -R kann man Zugriffsrechte einer Datei auch rekursiv ändern, d.h. Allen Leserechte an einem Dokument geben

```
$ chmod -R +r privat.doc
```

Um mehrere Rechte auf einem zu ändern könne Sie diese auch nacheinander schreiben und diese mit Komma trennen

```
$chmod a-wx, a+r mein Dokument.doc
```

Es gibt noch eine weitere Möglichkeit mit der Sie Zugriffsrechte im Terminal ändern können . Dies sieht auf den ersten Blick etwas komplizierter aus, ist es wenn man das Prinzip verstanden hat aber nicht. Man schreibt die Zugriffsrechte in Bit. Ein Zugriffsrecht besteht aus 3 Ziffern, von denen sich jeder auf eine Zugriffsebene (Benutzer, Gruppe, Alle) bezieht. Die Ziffern

entsehen aus der Addition folgender Werte:
4 = Lesen, 2 = Schreiben, 1 = Ausführen, 0 = keine Zugriffsrechte
Dies würde dann folgendermaßen aussehen:

```
$ chmod 644 privat.doc
```

644 gibt an, dass der Benutzer, der die Datei erstellt hat Lese- und Schreibrechte hat (4+2=6) und die Gruppe und Alle durch die „4" nur Leserecht an der Datei haben. Um dies noch deutlicher zu machen haben wir noch ein paar Beispiele dazu:

777 = rwxrwxrwx
775 = rwxrwxr - x
700 = rwx- - - - - -
444 = r - -r - -r - -

Das Sticky Bit
Wie vorher versprochen werden wir hier das Sticky Bit erklären. Wenn das Sticky Bit für ein Verzeichnis gesetzt wird, können Benutzer in dem Verzeichnis nur löschen, wenn sie der Besitzer dieser Datei sind, es genügt nicht, dass sie durch die Gruppe entsprechende Rechte hätten.
Das Sticky Bit wird auch wie ein Zugriffsrecht vergeben und erscheint am Ende dieser.
Dieser Befehl würde das Sticky Bit aktivieren

```
$ chmod u+t /meinOrdner
```

dies würde dann folgendermaßen aussehen
drwxrwxrwt 5 root admin 170 30 Nov .....

Benutzer
Da wir ja nun geklärt haben was Zugriffrechte sind und wie man diese ändert, sollten wir nun näher auf die Benutzer eingehen, den was nützen Zugriffsrechte, wenn man keine Benutzer hat?
Den wie in jedem UNIX-Multiusersystem ist auch Mac OS X die Benutzerverwaltung nicht wie bei den Mac OS X-Vorgänger eine Erweiterung, sondern eine wesentliche Funktion und daher auch tief verankert.

Was ist ein Benutzer ?
Ein Benutzer ist nicht unbedingt eine reale Person, sondern eine Einheit deren Datein gehören bzw. die Programme ausführen darf. Ein solcher Benutzer wird über ein Konto verwaltet. Damit das System den Überblick über die Benutzer behält, vergibt es für jeden Benutzer eine Benutzer-ID z.B. 504, als benutzer selbst sieht man diese ID nicht, sondern einen Benutzernamen, da ein solcher einfacher zu merken ist.

Gruppen
Jeder Benutzer gehört auch gleichzeitig einer oder mehrere Gruppen an. Das System benutzt Gruppen als Sammlung von Benutzern um diese als Einheit zu verwalten. Wie für einen Benutzer verwendet das system auf für Gruppen-IDs und als Ausgabe für den Benutzer Gruppennamen.
Als Beispiel: das System benutzt die Admin-Gruppe um die Benutzer zu verwalten, welche das system administrieren dürfen, so muss man nicht von Benutzer zu Benutzer immer wieder das selbe machen sondern kann die Eigenschaften über

die Gruppe vergeben.
Natürlich kann man die Benutzer- und Gruppen IDs am Terminal sichbar machen, dazu gibt man einfach folgenden Befehl ein

```
$ id
```

Die Zeile die Ihnen ausgegeben wir sieht in ihrer Art wie folgt aus

```
$ id
uid=501(Benutzer) gid=501(Gruppe)
groups=501(groups), 80(admin)
```

Besonders ist hierbei dass der Benutzer zusätzlich zur Gruppe „admin" gehört, das zeigt uns was nach dem Komma steht.
Nun wird Ihnen vielleicht aufgefallen sein, dass sie an Ihrem Rechner auch die ID501 haben. Dies ist kein Zufall, den Mac OS X verwaltet die Benutzer so, dass alle Benutzer ab ID=500 im Anmeldefenster gezeigt werden und alle darunter nicht sichtbar sind. Um die ID-s aller Benutzer zu sehen, muss man in die Netinfo-Datenbank schauen.

Benutzertypen
Es gibt unter Mac OS X drei Arten von Benutzern, die wir hier erläutern wollen.

Der „Normale" Benutzer
Normale Benutzer dürfen nur in ihrem eigenen Home-Verzeichnis tun und lassen was sie wollen, auf alle anderen Verzeichnisse, (wenn nicht nachträglich geändert) haben sie nur Lesezugriff.
Ein Administrator hat sogar die Möglichkeit Programme, das System oder den Finder für „Normale" Benutzer völlig anzupassen, so dass diese auch nichts im System durcheinander bringen können.

Der Administrator
Dieser ist normalerweise der Benutzer der das System installiert hat, allerdings können auch anderen Benutzern Administratorrechte zugewiesen werden. Der Administrator hat auch sein Home-Verzeichnis unter „Users" er hat aber viel mehr Rechte als ein normaler Benutzer.

root
Wenn man als „root" angemeldet ist hat man Vollzugriff auf das ganze System, denn der „root"-Benutzer ist ein Benutzer der nicht der Kontrolle durch die Zugriffsrechte untersteht. Dies ist allerdings mit Vorsicht zu geniesen, den auch alle Programme die von „root" gestartet werden haben „root"-Rechte, was im Falle eines defekten Programmes oder eines Viruses das System beschädigen kann. Deshalb sollte man sich als „root" nur wenn wirklich nötig anmelden.
Aus Sicherheitsgründen ist bei Mac OS X standardmäßig der Benutzer „root" deaktiviert. Um trotzdem einen „root"-Zugriff zu machen müssen Sie Administrator sein und das Programm „sudo" benutzen. Dazu zeigen wir wie man die Datei secure.log, die verschiedene Sicherheitssysteme von Panther enthält, einsehen kann.

```
$ sudo cat  /var/log/secure.log

Password: *******
```

um eine „Sudo“-Eingabe zu machen müssen Sie immer ihr root-Passwort eingeben, dann merkt sich „sudo“ dieses für einige Zeit, sodass Sie mehrere Befehle eingeben können, ohne bei jedem das Passwort erneut eingeben zu müssen.
Wenn Ihnen das immer noch zu wenig ist, gibt es die Möglichkeit eine Shell mit „root“-Privilegien zu öffnen

```
$ sudo -s
Password: ******
#
```

Ihr Prompt ändert sich nun auch von „$“ in „#“. Wie schon erwähnt bewegt man sich nun auf dünnen Eis und bei einem Fehler, kann man sein ganzes System zerstören. Für diesen Fall gibt es die Möglichkeit in die Datei `/var/log/system.log` zu schauen dort werden alle Befehle aufgezeichnet, dies kann eventuell behilflich sein um herauszufinden, was schief gelaufen ist.
Wie schon erwähnt ist bei Mac OS X aus Sicherheitsgründen der root-Benutzer deaktiviert. Dieser kann aber mit einem einfachen befehl aktiviert werden und ist nach Einrichtens des Passwortes aktiv:

```
$ sudo passwd root
```

Benutzer verwalten

Wie Sie im Linux-Teil lesen konnten, bzw. lesen werden, speichern diese Systeme genauso wie UNIX Systeme Ihre Benutzer- und Gruppeninformationen für gewöhnlich in den Dateien `/etc/passwd` und `/etc/group` – diese existieren unter Mac OS X, dieses benutzt diese aber nicht. Mac OS X verwendet zur Benutzerverwaltung die Open Directory. Dieser werden wir uns später genauer wittmen.

Benutzer anlegen

Um Benutzer nutzen zu können muss man dieser zuerst logischerweise anlegen und hier kommen wir zu einer ganz komischen Sache, den dies ist ein Punkt an dem wir nicht mit der Kommandozeile arbeiten können, da Apple ein Tool wie „usserad“ einfach weggelassen hat, aus welchen Gründen konnten wir bislang leider noch nicht erfahren und so müssen auch wir zur Benutzerverwaltung die GUI und dort die Systemeinstellungen/Benutzer/.. wählen. Hier können nun alle benutzerrelevanten Daten angelegt werden. Wie das im einzelnen Funktioniert wollen wir an dieser Stelle nicht erklären, da wir uns nicht so lange an der GUI aufhalten möchten.

Netinfo Manager

Es gibt noch eine weitere Möglichkeit Benutzer zu verwalten, der Netinfo Manager. Dieser gewährt Einblick in die Netinfo-Datenbank, in welche alle angelegten Benutzer von der Open Directory eingetragen werden.

Nicht-menschliche Benutzer

Auch wenn Sie der einzigste menschliche Benutzer an Ihrem Rechner sind, gibt es noch einige mehr Benutzerkonten darauf. Allerdings sind die meisten dieser Konten nicht für den Gebrauch von menschlichen Benutzern gedacht sondern für den Gebrauch durch verschiedene Programme und Dienste des Systemes angelegt und diese nennen sich nicht-menschliche Benutzer.

Diese haben z.B. den Vorteil, dass ein Apache-Webserver in einer kontrollierten Umgebung laufen kann, sodass bei einem Hacker-Zugriff über diese, der mögliche Schaden begrenzt ist.
Wir wollen hier die nicht-menschlichen Benutzer auflisten und erläutern.

Name	ID	Beschreibung
root	0	Administrator-Benutzer
daemon	1	Kern-System-Daemons
smmsp	25	Sendmail-Benutzer
lp	26	Druckdienste
postfix	27	Postfix-SMTP-Server
www	70	Apache-Webserver
eppc	71	Apple-Events
mySQL	74	MySQL-Datenbank (OS X S)
sshd	75	SSH-Server-Daemon
qtss	76	QT-Streaming-Server (OS X S)
cyrus	77	POP-, IMAP-Server (OS X S)
mailman	78	Mailman (M-Listen) (OS X S)
appserver	79	JBoss App-Server(OS X S)
nobody	-2	Benutzer mit wenig Rechten

Die mit „(OS X S)“ gekennzeichneten Dienste sind nur bei der Mac OS X Server-Version vorhanden.

Die Systemaktivität
Nun haben wir bei Grundstein bezüglich Dateien, Benutzer und Betriebsystem gelegt. Jetzt sollten wir uns einmal genauer anschauen was unter System eigentlich macht. Unter der GUI gibt es dazu die Konsole und die Aktivitätsanzeige, die sind ganz gut aber wir wollen uns auch hier wieder des Terminals bedienen das da wohl auch noch ein wenig bessere Werkzeuge hat.

Protokolldateien
Es gibt verschiedene Ort, wo sich Protokolle befinden können:

~/Libray/Logs/
Hier sind die zu einem Benutzer gehörenden Protokolle. Hier liegen z.B. die Protokolle des Crash-Reporters, der die Programmabstürze aufzeichnet

/Library/Logs/
Hier liegen die Protokolle die das gesamte System betreffen, z.B. der CrashReport des abgestürzten Systemkomponenten. Hier wären auch die Protokolle für den Fall einer Kernel-Panic

/var/log
Hier sind die Protokolle der verschiedenen auf Ihrem System laufenden BSD-Unix-Utilities. Hier sind z.B. die Protokolle des Appache-Webservers und der ipfw-Firewall.

Arbeiten mit Protokollen
Um mit den Protokollen auf der Kommandozeile zu arbeiten gibt es die beiden Tools „grep“ (get regular expression“ und „tail“. „grep“ ist für die

Suche und Ausgabe von Textzeilen einer Datei da. Als Beispiel wollen wir mal nach dem Begriff CHUD in den Systemprotokolldateien suchen, dies funktioniert folgendermaßen

```
$ grep CHUD /var/log/system.log
```

Nun wird Ihnen die Kommandozeile alle Zeilen mit dem Begriff „CHUD" darstellen. Mit „grep" kann man aber auch nicht nur nach einem Begriff suchen, sondern reguläre Ausdrucke (regex) zur Suche verwenden. Wenn wir z.B. nach alle Zeilen suchen möchten die den Begriff „CHUD" und gleichzeitig den Begriff „Kernel" erhalten, machen wir das so

```
$ grep CHUD.*kernel /var/log/system.log
```

So kann man ganz einfach Bedingungen erstellen um seine Suche einzugrenzen.
Da man in einem Protokoll meist die aktuellsten Einträge interessant sind, diese aber immer am Ende der Datei stehen gibt es den Befehl „tail" Ein Befehl könnte dann so aussehen

```
$ tail /var/log/system.log
```

Nun sehen Sie die letzten zehn Zeilen des Protokolls. Sie können den Befehl „tail" durch das Attribut „-n" beeinflussen um eine andere Zeilenzahl darzustellen

```
$ tail -n1 /var/log/system.log
```

zeigt Ihnen die letzte Zeile des Protokolls. Ein weiteres Argument ist „-f", damit sind Sie wie Premiere sagen würde „Mittendrin statt nur dabei", den so können Sie in das Protokoll schauen während es geändert wird

```
$ tail -f -n0 /var/log/system.log
```

Prozesse

Mit dem Befehl „ps" für Prozessstatus kann man in der Kommandozeile Informationen über verschiedene Befehle darstellen. Mit dem Befehl **„ps"** erhalten Sie den Überblick über den von Ihnen gestarteten Shell-Prozess, wenn Sie die Prozesse sehen möchten, die Ihnen gehören dann müssen sie das Attribut **-x**

```
$ ps -x
```

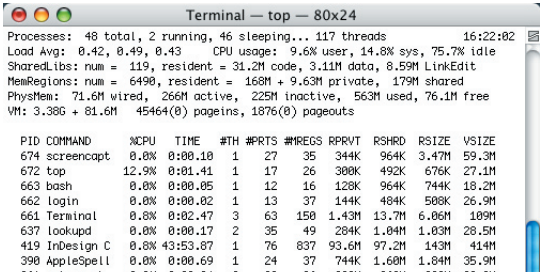
anfügen. Wenn Sie einen bestimmten Prozess suchen könne Sie auch hier das Tool „grep" anwenden.

```
$ ps -x | grep Photoshop
```

dann werden Sie Ihren Photoshop finden. Um alle Prozesse, die auf Ihrem System laufen darzustellen benötigen Sie das Attribut „ **-aux**", da dies allerdings sehr viele sind ist es ratsam die suche mit „grep" einzugrenzen.

Aktivität der Prozesse

Um zu sehen wie die einzelnen Prozesse Ihren Prozessor auslasten oder andere Aktivitäten zu beobachten, müssen sie den Befehl „**Top**" im Terminal eintippen. Um dies zu beenden wählen Sie „Apfel" + „.". Dies sieht dann so aus:



Prozesse beenden

Wenn sich nun ein Prozess „aufgehängt" hat und man diesen abrechen möchte, gibt es in der Kommandozeile den Befehl „kill" um dies zu erledigen. Dazu muss man mit „ps" die Prozess-ID herausfinden und diese dann wie folgt eingeben (z.B. Prozess-ID 578)

```
$ kill 576
```

wenn es vorkommen sollte, dass dieser Befehl seine gewünschte Wirkung nicht ganz erfüllt, dann kann man diesen mit dem Attribut -KILL noch verstärken.

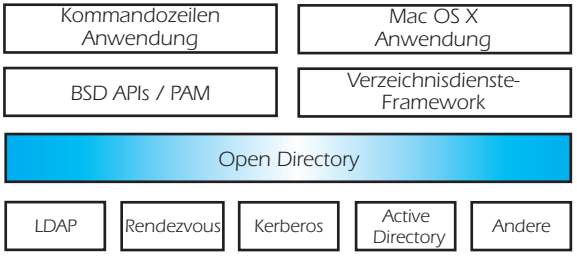
```
$ kill -KILL 576
```

Die Open Directory

Wie schon weiter vorne erwähnt möchten wir uns nun der Open Directory wittmen.
Die Open Direcotry spielt eine ganz zentrale Rolle bei Mac OS X. Sie ist zuständig für die Verwaltung und Bereitstellung von Benutzern, Rechnern und Druckern. Was sich die meisten Benutzer gar nicht bewusst sind, ist dass sie schon beim Anmelden an Ihrem Rechner oder beim Verbinden mit einem Server die Open Directory benutzen.
Um genauer zu verstehen was die OpenDirectory macht werden wir hier zeigen was geschieht, wenn Sie ihr Benutzernamen und das Passwort bei der Anmeldung eingegeben haben.
1. Das Anmeldefenster frägt OpenDirectory nach der Authentifizierung des Benutzers.
2. Open Directory prüft den Benutzernamen und das Passwort und sucht nach der Authentifizierungsmethode , falls der Benutzer existiert
3. Open Directory versucht das Passwort zu verifizieren
4. Open Directory teilt dem Anmeldefenster mit, ob der Benutzer authentifiziert wurde oder nicht
5. Wenn der Benutzer authentifiziert wurde, startet das Anmeldefenster die GUI, während sie die Open Firmware nach dem Ort des Homeverzeichnisses des Benutzers frägt.
Alle Systemteile befolgen diese Vorgang. Manche von Ihnen greifen aber nur indirekt auf die Open Firmware zu und zwar über das sogenannte PAM (pluggable authenication modul), welches in vielen Unix-Anwednungen integriert ist.
Um dies deutlicher zu machen zeigen wir noch die Authentifizierung bei einer ssh-Sitzung, den

da kommt PAM zum Einsatz.
1. sshd erhält Benutzernamen und Passwort von dem, der sich anmelden möchte.
2. sshd startet einen PAM-Aufruf um den Benutzer zu authentifizieren. PAM spricht direkt die Open Firmware an.
3. Diese prüft nun wieder Benutzetnamen und Passwort und gibt die Informationen direkt an sshd zurück.
Ein weitere Funktion der Open Directory ist die Netzwerkverbindung. In dem Moment, in dem sie im Terminal „Apfel + K" eingeben, sprechen Sie die Open Directory an, welche dann versucht mit Hilfe eines seiner Protokolle die Informationen darzustellen.
Die Open Directory arbeitet mit Verzeichnisdiensten, daher sollte dieser Begriff zuerst geklärt werden:

Verzeichnisdienste
Wie Sie im Teil Linux lesen konnten werden in UNIX Systemen die Benutzerinformationen, -passworter in Textdatein unter /etc/... gespeichert.
Das funktioniert bei einem Rechner auch ganz gut, aber sobald man Benutzer von mehreren Rechner dazu noch Drucker und Server verwalten muss, wir es ganz schon umständlich und die Entwickler haben sich an Unterschiedliche Lösungen dieses Problemes gemacht.
SUN entwickelte NIS, Novell NetWare, welche sich dann Microsoft angenommen hatte und es in Active Directory wandelte und NEXT entwickelte Netinfo, welches Apple dann für Mac OS X einsetzte. Apple wusste aber dass es nicht mehr genüge mit Mac-Netzwerken zu arbeiten und entwickelte weitere Technologien, welche dann gemeinsam den Namen Open Directory erhielten. Apple ist es so möglich folgende Technologien zu unterstützen: LDAP (Lightweight Acces Diretoty Protokoll), Kerberos und Active Directory.



In dieser Grafik ist zu sehen, wie Max OS X Anwendungen direkt auf die Open Directory zugreifen können, damit man von der Kommandozeile aus auch immer auf die selbige zugreifen kann, wurden Authentifizierungsroutinen so verändert, dass diese immer auf PAM und diese dann auf die Open Directory zugreifen.

Informationen der Open Directory
Hier möchten wir ihnen die Informationen zeigen mit denen die Open Directory umgeht
- Informationen zur Benutzerverwaltung (Benutzer-ID und der Benutzername)
- Informationen zur Authentifizierung (Benutzerpasswörter)
- Informationen zu Gruppen
- Die Angaben zu Home-Verzeichnissen der Benutzer
- Netzwerk-Dateisysteme

und Programmen
- Informationen zu Netzwerkdrucker

Nun wird wohl auch klar wieso die Open Directory eine so große Rolle beim System Mac OS X spielt, diese von der Open Directory verarbeiteten Informationen werden gerne in den Begriffen Konfiguratioinsinformationen (LDAP, Netinfo, Active Directory und NIS), Authentifizierungsdienste (Shadow-Passwortdatenbank – /var/db/shadow/, Apple Passwort Server - nur OS X Server, Kerberos und Active Directory) und erkannte dienste (Rendezvous, SMB, SLP und Apple Talk), zusammengefasst.

Verzeichnis-Domains

Obwohl es immer mehr netzwerkbasierende Verzeichnisdienste gibt, benötigt man die lokalen Verzeichnisdienste immer noch, sonst könnte man ein Rechner nicht mehr ohne Netzwerkdienste betreiben. Dass das Betriebsystem dabei aber nicht durch einander kommt, hat man das Problem beim Mac OS X mit Domains gelöst.

Lokale Domains

Diese wird standartmäßig bei Mac OS X mitinstalliert, diese wird auch zuerst von der Open Directory befragt

Shared Domains

Dies sind eine beliebige Anzahl weiterer netzwerkbasierenden Domains, die konfiguriert werden müssen um an einer Netzwerkumgebung teilnehmen zu können.

Die lokale Domain und Netinfo

Die lokale Domain besteht aus folgenden Teilen:
- Die lokale Netinfo Datenbank, diese liegt unter /var/db/netinfo
- Die lokale Shadow-Datenbank, diese befindet sich unter /var/db/shadow
- Rendezvous, SLP und SMB zur Erkennung von Netzwerkdateisystemen

Die Shadow-Datenbank

Da diese eine Neuerung in Mac OS X Panther ist, wollen wir auf diese speziell eingehen.
In den Vorgänger Mac OS X Versionen wurden die Passwörter mit dem Kommandozeilen-Tool „crypt“ verschlüsselt und direkt in der Netinfo-Datenbank gespeichert. Diese konnten von dort aus ausgelesen und „gehackt“ werden.
Bei der Shadow-Datenbank sind die Einträge nun in einem Verzeichnis gespeichert, dass nur durch „root“ gelesen werden kann.
Die Netinfo-Datenbank kann mit dem Netinfo-Manager, den wir ja schon kennengelern haben modifiziert werden.
Da unser es unser vorhaben ist „unter die Haube“ von Mac OS X zu sehen, wollen wir nun die Werkzeuge, die es auf der Kommandozeile gibt, um mit der Open Directory zu arbeiten, wittmen.
Kommandozeilenwerkzeuge
Wir werden hier einen Überblick über die Werkzeuge geben und im Anschluss den Umgang mit den wichtigsten davon erklären.
Bevor man damit loslegt sollte man sich ein Backup der Netinfo-Datenbank erstellen, damit man keinen großen Schaden anrichtet.

Sichern der Datenbank
Eine Backupdatei der Datenbank erstellt man mit folgendem Befehl

```
$ sudo cp -R /var/db/netinfo/local.nidb /var/backups/backup.nidb
```

Dieser Befehl kopiert die Datenbank an in einen Backup-Ordner mit dem Namen „backup“. Wenn es nun wirklich soweit gekommen ist, dass die Netinfo-Datenbank defekt ist, dann muss man den Rechner im Single-User-Modus booten (Apfel + S) und folgende Befehle eingeben

```
$ /sbin/fsck -y
```

um das Dateisystem auf Fehler zu prüfen

```
$ /sbin/unmount -uw /
```

um die Festplatte im Lese-, Schreibmodus zu mounten

```
$ mv /var/db/netinfo/local.nidb /var/backups/damaged.nidb
```

um die defekte Datenbank zu verschieben

```
$ cp -R /var/backups/backup.nidb /var/db/netinfo/local.niidb
```

um die Sicherung an den richtigen Ort zu kopieren, nach einem Neustart sollte wieder alles ordnungsgemäß funktionieren. Und nun wollen wir mit den Werkzeugen fortfahren

dscl
Ein Werkzeug für den Umgang mit den Daten-Quellen der Open Directory (z.B. LDAP, SMB)

nicl
Ein Werkzeug zur bearbeitung der Netinfo-Datenbank

nidump
Zum Laden von Dateien aus der Netinfo-Datenbank

niload
Zum Laden von Dateien in die Netinfo-Datenbank

niutil
Ein Werkzeug zum Lesen und Bearbeiten der Netinfo-Datenbank

nifind
Um die Netinfo-Datenbank nach Verzeichnissen mit einem Muster zu durchsuchen

nigrep
um die Netinfo-Datenbank nach Verzeichnissen mit einem bestimmten Muster zu durchsuchen.

dscl
dscl ist ein Tool, das gestartet wird und dann solange läuft bis es mit einem „quit“ beendet wird. Gestartet wird so

```
$dscl .
```

Ihre Eingabeaufforderung sieht dann folgendermaßen aus und erwartet einen Befehl von Ihnen

```
$ dscl .
/<
```

nun können wir ja mal ein Blick in die Netinfo-Datenbank werfen, indem wir uns zuerst mit dem Befehl „`list`“ eine Auflistung anzeigen lassen.

Und mit dem Befehl „`cd Users`“ das Verzeichnis Users einsehen. Dies ist nun das Selbe wie wir im Netinfo-Manager auf der Kommandozeile sehen können.
Wer nun hier weiterarbeiten möchte kann sich mit dem Befehl „`help`“ die gewünschten Befehle für „dscl“ suchen.

nicl
Der Befehl „nicl“ ist ziemlich ähnlcih wie dscl, er wir auch mit

```
$nicl .
```

aktiviert und mit „`quit`“ beendet.
mit dem Befehl „list“ könnne wir uns die Verzeichnisse in der netinfo-datenbank anzeigen lassen.
Wer auch hier weiterarbeiten möchte kann sichmit dem Befehl „help“ die entsprechenden Befehle heraussuchen.

nidump
Wie schon erwähnt ist „nidump“ ein Befehl um die Inhalte der Netinfo-Datenbank auszugeben, die ist in der Unix-Welt zum erstellen eines Backups gedacht un deshalb wollen wir dazu den befehl liefern

```
$ nidump -r / . > nibackup.txt
```

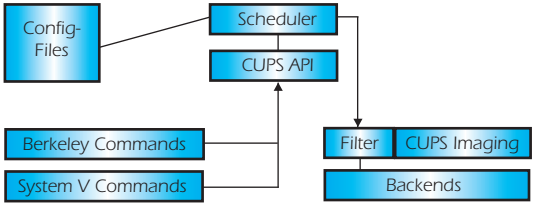
niload
„niload“ dient nun dazu das erstellte Backup wieder in die datnbank zurück zugeben

```
$ niload -r / . <nibackup.txt
```

Nun das war auch soweit dass was wir an Befehlen zur Manipulation der Open Directory geben wollen. Wir wollen einen Einstieg darin geben und überlassen alles andere den Entwickler dieser Welt,die sich damit richtig auskennen und über dieses Thema mit Sicherheit eine weitere Ausarbeitung schreiben können.

Drucken
In diesem Abschnitt wollen wir ein Einblick in das Drucken mit Mac OS X geben, aber wie Sie es ja schon gewohnt sein werden wollen wir uns wieder auf das Drucken mit der Kommandozeile und dessen was dahinetr steckt beschäftigen.

CUPS
Früher gab es unter UNIX zwei Drucksysteme, das LPD von BSD und das System V LP. Da diese Systeme ursprünglich für die Ansteuerung von Zeilendruckern konzipiert waren, könnten damit moderner Drucker entweder gar nicht oder nur teilweise angesteuert werden.
Es gab in den vergangenen Jahren immer verschiedene Ansätze der Druckerindustrie dies Problem zu beheben, mit mehr oder weniger Erfolg. Eine neue plattformübergreifende Technik musste her.
Ein loser Verband von Druckerherstellern (PWG) definierte 1999 das Internet Printing Protokoll (IPP), welches auf dem weitverbreiteten Standart HTTP aufsetzt. IPP erlaubt die Steuerung und Kontrolle von Druckjobs sowie Authentifizierungen und Verschlüsselungen. Die älteste und wohl auch verbreiteste Implementierung von IPP ist CUPS, welches auchunter Mac OS X Verwendung findet.
Die nachfolgende Abbildung erklärt den prinzipiellen Aufbau von CUPS.



Die sogenannte Zentrale ist der Scheduller, dieser nimmt die Druckjobs über die CUPS API entgegen, holt sich die nötigen Informationen aus den Config-Files und gibt diese an die Filter, welche die Daten für den entsprechenden Drucker aufbereiten, meist als PS oder PDF und schickt diese dann zu den Druckern.

Druckmanager und Cups

CUPS ist aber nicht alleine für das Drucken unter Mac OS X verantwortlich. Was beim Drucken von Mac OS X aus so besonders ist wollen wir hier erleutren. Mac OS X hat einen eigenen Printjob-Manager, der seit Panthereng mit CUPS verbunden ist. Dies hat den Vorteil, dass man egal ob man von der GUI oder CLI aus druckt, die selbe auswahl an Warteschlangen und Treibern hat. Der Printjobmanager nimmt die Jobs an und gibt diese in Form von Post-Script oder PDF an Cups weiter der große vorteil darin ist, dass der Printjob-Manager auch ppds (Druckerbeschreibungsdateien) an CUPS weitergeben kann, sodass die Funktionalität des Druckers in vollem Umfang genutzt werden kann. Apple hat sich auch noch Gimp-Print, eine Sammlung von Open Source Druckertreiber ins Boot geholt um damit die bestmögliche Auswahl an Druckertreibern bieten zu können und damit allen anderen mal wieder ein Schritt voraus zu sein.

Drucken mit der Kommandozeile

Nun öffenen wir unser Terminal und wolenuns zerst mal anschaun welche Drucker mit unserem Rechner verbunden sind

```
$ lpstat -a
```

```
68          afpuser_aliases
/ > quit
Goodbye
Daniel-Windelbands-Computer:/ danielwindelband$ lpsat -a
-bash: lpsat: command not found
Daniel-Windelbands-Computer:/ danielwindelband$ lpstat -a
_192.168.2.5 accepting requests since Jan 01 00:00
DESKJET_920C accepting requests since Jan 01 00:00
Drucken_in_Datei accepting requests since Jan 01 00:00
Stylus_COLOR_760 accepting requests since Jan 01 00:00
Daniel-Windelbands-Computer:/ danielwindelband$
```

so oder so ähnlich, wird die Ausgabe in Ihrem Fesnter zu sehen sein. Wenn ein Drucker erkannt wird der über einen anderen Rechner angesteuert wird, steht nach dem namen noch der name des rechners z.B. HP1050@andererRecher. Um herauszufinden welcher Drucker als standarddrcuker angelegt ist, geben sie den Befehl

```
$ lpstat -d
```

ein

```
$ lstat -d HP 1050
```

würde unseren HP 1050 zum Standaddrucker machen.
Da wir nun unseren Standarddrucker festgelegt haben, können wir uns jetzt ans eigentliche Drucken machen. das Tool, dass unter UNIX zurm Drcuken verwedet wird ist „lpr" und wir folgendermaßen angewandt

```
$ lpr „option“ „Dateiname“
```

Es gibt einige Optionen aber als sinnvolle erschin uns diese: „-P…“ „…“ hier wird der Names des Druckers einegsetzt, auf dem gedruckt werden soll.
nach dem „Wegdrucken“ des Druckjobs, erscheint wieder ein Prompt, d.h. aber nicht, dass der Druckjob weggedruckt ist. Um dies zu sehen, können Sie aber die Druckerwarteschlange einsehen, dies macht der Befehl

```
$ lpq
```

Nun sehen Sie die Druckerwarteschlange, in der erten Zeile ist der Status des Druckers zu sehen und darunter die Jobs, die noch anstehen.
Mit dem Befehl

```
$ lprm „Job-ID“
```

können Sie den entsprechenden Druckjob abbrechen, die Job-ID sehen Sie wenn Sie „lpq“ eingeben.

Apple Talk Drucker

Wenn Sie sich in einem Netzwerk befinden, in dem sich Apple-Talk Drucker beifnden, die sie ansprechen möchten, müssen Sie erst mal den Drucker mit folgendem Befehl ausfindig machen

```
$ atlookup
```

Nun sehen Sie die Drucker die Ihnen zur Verfügung stehen. Nun können Sie mit dem Befehl

```
$ atprint
```

ihre Datei an den Drucker senden.
Wichtig hierbei zu wissen ist, dass die meisten Appletalk-Drucker Post-Script-Drucker sind und deshalb PS- oder PDF-Dateien zum Druck benötigen. um eine Post-Script-datei auf einem Apple-Talk-Drucker auszugeben, benötigens Sie folgenden Befehl

```
$ cat Dokumnet.ps | atprint
```

Dieser Befehl sollte aber noch kurz erklärt werden. Mit „cat“ offnen Sie die zudruckende Datei und senden Sie über „|“ an den Apple-Talk-Drucker

Post-Script

Im Abschnitt Apple Talk-Drucker haben wir gesagt, dass Sie eine Postscriptdatei benötigen um auf einen Apple-Talk-Drucker zu drucken, aber was tun, wenn Sie diese nicht haben ? dazu gibt es auf der Kommandozeile auch ein ein Tool, dieses nennt sich „enscript“ und wird folgendermaßen benutzt

```
$ enscript -p „zielname“.ps „zu konvertierende Datei“
```

wie Sie sehen ist auch dies auf der Kommandozeile kein Hexenwerk. Sie müssen einfach den Zielnamen mit „.ps“ und dahinter den Dateinamen der zu konvertierenden Datei.

PDF

Da es nicht viel schwerer ist eine PDF Datei aus einem Post-Script-File zu erzeugen, möchten wir das hier auch noch zeigen. der Befehl dazu lautet

```
$ pstopdf „Dateiname.ps“
```

einfach für dateiname, den name der Post-Script -Datei angeben, die sie kovertieren möchten.

Netzwerkverbindungen

da wir ja nun schon einige Dinge über unseren Rechner gelernt haben, wollen wir Ihnen nun zeigen wie Sie mit Ihrem Rechner in einem Netzwerk verkehren können. Die Grundlage dazu ist es sich eine IP-Adresse einzurichten.
Dazu geben Sie imTerminal folgenden Befehl ein

```
$ ifconfig
```

```
Terminal — bash — 130x
Daniels-book:~ danielwindelband$ ifconfug
-bash: ifconfug: command not found
Daniels-book:~ danielwindelband$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 ::1 prefixlen 128
        inet6 fe80::1 prefixlen 64 scopeid 0x1
        inet 127.0.0.1 netmask 0xff000000
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::20a:95ff:feaa:92c prefixlen 64 scopeid 0x4
        inet 172.16.1.102 netmask 0xffff0000 broadcast 172.16.255.255
        ether 00:0a:95:aa:09:2c
        media: autoselect (100baseTX <full-duplex>) status: active
        supported media: none autoselect 10baseT/UTP <half-duplex> 10baseT/UTI
 100baseTX <half-duplex> 100baseTX <full-duplex> 100baseTX <full-duplex,hw-loo
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:0d:93:eb:a1:6e
        media: autoselect (<unknown type>) status: inactive
        supported media: autoselect
fw0: flags=8822<BROADCAST,SMART,SIMPLEX,MULTICAST> mtu 2030
        lladdr 00:0a:95:ff:fe:aa:09:2c
        media: autoselect <full-duplex> status: inactive
        supported media: autoselect <full-duplex>
Daniels-book:~ danielwindelband$
```

Hier sehen Sie nun die Netzwerkkonfiguration Ihres Rechners „Lo0“ steht für Ihre localhost-Adresse. „en0“ für Ihre erste Netzwerkkarte, diese zeigt die IP-Adresse aber nur, wenn Sie ein Netzwerkkabel eingesteckt haben. In diesem Beispiel ist noch eine „en1“ zu sehen, dies liegt daran, dass in diesem Rechner eien Airport-Karte installiert ist.

Remote Login

Manchmal ist es ja so, dass man auf seinen Rechner zugreifen möchte, aber nicht vor diesem sitzt, weil man gerade in einem anderen Büro etwas zu tun hat, oder jemand anderes an Ihrem Rechner arbeitet. Für solche Fälle gibt es das sogenntte Remote-Login. Das heißt, dass man sich von einem anderen Rechner aus, über das Netzwerk auf seinem Rechner einloggen kann um seine Arbeit zu verrichten. Einen solchen Remote-Login macht man eigentlich auch nur von der Kommandoteile aus, da man sonst grafische Informationen mit übertragen müsste was die Verbindung unnötig langsamer machen würde. das Tool dass auch bei Mac OS X auf der Kommandozeile für den Remote-Login gibt nennt sich „ssh“. Um diese sogenannte Secure-Shell zu verwenden, müssen wir unsleider der GUI bedienen und unter Systemeinstellungen/Sharing/ den Dienst „entfernte Anmeldung“ aktivieren.

ssh

Wenn sie nun den Dienst „ssh“ gestertet haben, besteht für Sie jetzt die Möglichkeit sich von einem entfernten Rechner aus einzuloggen.
Was Sie dazu auf jeden Fall wissen müssen, ist die IP-Adresse, Benutzername und Passwort Ihres Rechners. Für unser Beispiel nehmen wir mal an wir hätten die IP-Adresse: 192.168.1.107 und den Benutzernamen Max, dann würde der Remote-Login von einem anderen Rechner auf unseren so aussehen

```
$ ssh -l max@192.198.1.107
```

Sie müssen mit „yes“ noch bestätigen, dass Sie sich anmelden wollen und Ihr Passwort eingeben. Schon haben Sie eine sogenannte ssh-Sitzung zu Ihrem Rechner aufgebaut und können damit arbeiten.

ftp

Um Daten zwischen zwei Rechnern auszutauschen, wir meist FTP (File-Transfer-Protokoll) verwendet. Dazu müssen Sie in der Systemeinstellung den Dienst „FTP“ aktivirenen und können dann mit folgendem Befehl auf Ihrer Rechner zugreifen
$ ftp „Rechnermane“
„Rechnername“ muss durch den Namen Ihres Rechners ersetzt werden. Wenn Sie verbunden sind können Sie nun Dateien auf den rechner kopieren, dies machen Sie mit dem Befehl „put“

```
$ put Brief.doc
```

und um Dateien herunterzulanden get

```
$ get Brief.doc
```

Das sind eigentlich die beiden wichtigsten Befhele um mit FTP umzugehen.

Alltagstipps

In diesem Kapitel möchten wir Ihnen einige nützliche Tipps geben, die Ihnen das Leben mit Mac OS X leichter machen, bzw. Ihnen dabei helfen sollen Probleme mit Ihrem System auszumerzen. Auch hier wollen wir uns wieder darauf Konzentrieren, wie Sie Probleme mit dem Betriebsystem lösen können, dabei werden wir auch zum ersten mal die GUI und die CLI gleichberechtigen, da bei Problemen doch „alle zusammenhelfen" sollten.

Das Schriftenproblem

Vielleicht könnten Sie es mit der Zeit beobachten, wie Ihr Systemstart immer länger dauert, besonders wenn Sie schon die GUI sehen können, aber der „Sat1-Ball" sich noch munter dreht, und das minutenlang.
Dies liegt in den meisten Fällen einfach daran, dass zu viele Schriften in der Schriftenverwaltung geladen und auch noch aktiviert sind. Deshalb ist es zu empfehlen, dass man nur die Schriften lädt die wirklich benötigt werden, wenn man nicht enorme Performanceeinbusen in Kauf nehmen will.

Nicht-startende Programme

Das Problem, das die meisten User nach bereinigen der Schriftenverwaltung haben ist, dass Programme wie das z.B. Adressbuch sich nicht mehr öffnen lassen, dies ist darauf zurückzuführen, dass Systemschriften deaktiviert wurden, was zur Folge hat, dass Programme einfach nicht mehr Starten.
Nach Aktivierung der Systemschriften, funktionieren die Programme wieder einwandfrei.

Rechtestruktur wiederherstellen

Ein häufiges Problem bei Mac OS X ist, dass die Rechtestruktur beschädigt ist. Dies macht sich bemerkbar, wenn Programme oder Bestandteile des Bertiebsystem nicht mehr richtig funktionieren, oder nicht startet.
Um die Rechtestruktur wiederherzustellen, starten Sie das Festplattendienstprogramm, welches sich unter Programme /Dienstprogramme befindet. Hier gibt es dann die Funktion „Volume-Zugriffsrechte reparieren" wenn Sie sich nicht ganz sicher sind, ob dies Ihr Problemn ist, dann könne Sie zuvor die „Volume-Zugriffsrechte überprüfen".

Festplatte reparieren.

Wer im Frstplattendienstprogramm aufmerksam gewesen ist, hat festgestellt, dass sich das Volume hier auch überprüfen und reparienen lässt. Dies ist ja schön und gut, aber was macht man wenn der Rechner erst gar nicht hochfährt und man das Festplattendienstprogramm gar nicht starten kann ?
Auch hier muss man nicht gleich die Flinte ins Korn werfen. Man startet den Rechner im „Single-User-Modus" (dieser ist weiter vorne erklärt). Nun kann man „fsck" bei aktiven JFS „fsck -f" ein und der Rechner repariert die Platte. Wenn dieser Vorgang abgeschlossen ist, kann man den Rechner mit einem „reboot" neu starten.

Löschen des NVRAM

Eine weitere Möglichkeit um Ihr system zu reparieren, ist das Löschen des NVRAMs bekannt auch unter Parameter RAM. Um dies zu tun müssen SIe bei Start Apfel + Alt + P +R gedrückt halten und

warten bis der Startton ertönt, dies müss man oft 3 - 4 mal durchführen, bis der NVRAM wirklich gelöscht ist.

Open Firmware zurücksetzen

Wenn auch dies Ihren Rechner noch nicht in den gewünschten Zustand gestezt hat, dann gibt es noch die Möglichkeit die Open Firmware zurückzusetzen und damit, die nachträglichen (gewünschten oder nicht gewünschten) Konfigurationen des Rechners zu löschen und diesen auf Werkseinstellunng zurücksetzen. Um die Open Firmware zurückzusetzen müssen Sie Apfel + Alt + O + F drücken – nun befinden Sie sich in der Open Firmware. Jetzt müssen Sie folgende Befehle eigeben

```
reset-nvram

set-defaults

reset-all
```

da der NVRAM Bestandteil der Open Firmware ist, wird dieser hier noch mal mit gelöscht.

Unix-Wartumgsarbeiten

Da Mac OS X wie ja nun bekannt sein müsste auf UNIX bassiert, versucht das system Nachts einige Routine-Wartungsarebieten durchführen. Da viele Rechner über nacht ausgeschalten sind bleiben diese Wartungsarbeiten (dazu gehören Sicherung und Aktuallisierung der Netinfo-datenbank) auf der Strecke und es empfiehlt sich, dass Sie diese manuel hin und wieder durchhführen. dazu benutzen wir wieder unser Terminal. Dort geben wir folgenden Befehl ein:

```
sudo peridodic daily weekly monthly
```

damit führen Sie mit dem Tool „periodic" die täglichen, wochentlichen und monatlichen Aufgaben aus.

Rücksetzen des Displays bei Power- und i-books.

Teilweise ist es vorgekommen, dass nach einen Systemupdate, das display eines mobile rechners nicht mehr einwadfrei funktioniert. Dies liegt an einem fehler der Helligkeitssteuerung, die es eigentlich n ur beim Powerbook gibt, um diesen Fehler zu beheben, hilft es entweder den NV-Ram zu löschen,. ,oder während des Starts „R" gedruckt halten, das setzt das Display auf Werkszustand zurück.

Passwort vergessen

Wenn Sie Ihr Passwort vergessen haben, haben Sie folegnde Möglcihkeit, Sie müssen von der System-CD starten, dann können Sie im Insatller Benutzerpasswörter anlegen und ändern.

Quellen

Um diese Ausarbeitung fertig zu stellen haben wir uns folgender Quaellen bedient

Internet:

www.apple.com/de/osx – 01.2005 – 03.2005
www.thinkmacoxs.ch – 01.2005 – 03.2005
www.andre-aulich.de – 01.2005 – 03.2005

Bücher:

Mac OS X Panther – O`Reilly Verlag
UNIX für Mac OS X Panther – O`Reilly Verlag
UNIX in a Nutshell – O`Reilly Verlag
Linux – Kernelprogrammierung – Addison-Wesley
Linux/Unix – Grundlagen – Addison-Wesley