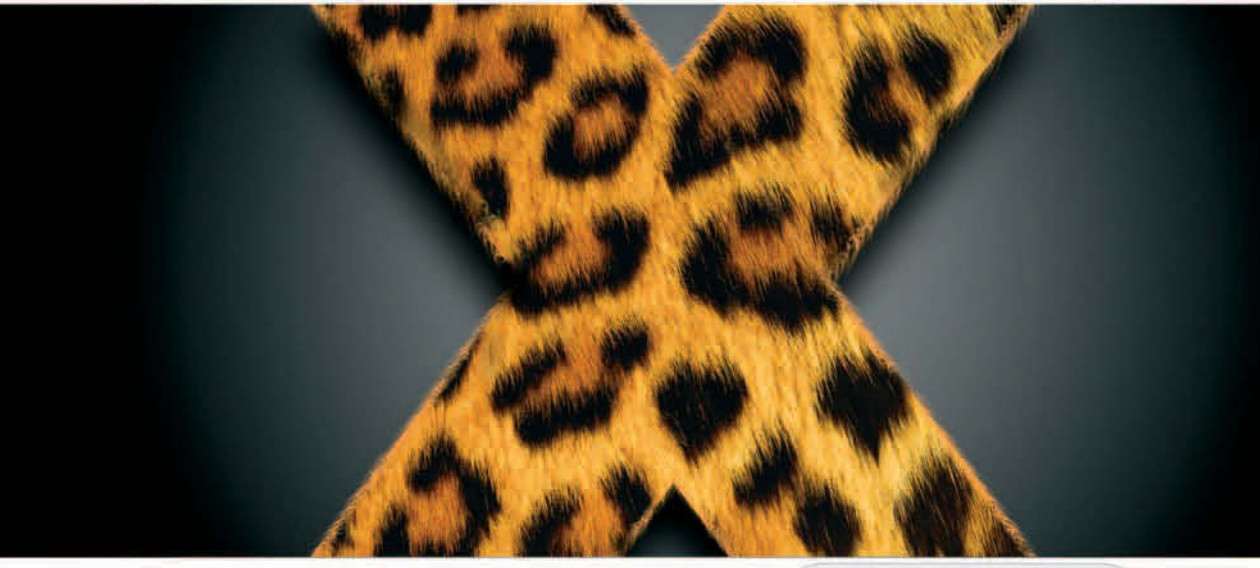
Uthelm Bechtel

Apple Mac OS X 10.5 Leopard

Der PowerFinder für Ihren Mac



für G4-, G5- und Intel-Mac

ADDISON-WESLEY

Technik

Anatomie

Technisches zur Hardware der Macs

Die Prozessoren

Zum Betrieb eines Computers werden verschiedene Komponenten gebraucht. Die wichtigsten Komponenten dabei sind der Prozessor und der Chipsatz.

► 32 Chipsatz

Der Prozessor ist die zentrale Recheneinheit eines Computers. Er führt Befehle aus, nimmt Berechnungen vor und verteilt Aufgaben an die verschiedenen Subsysteme.

Das neue Logo der Power.org, einem Zusammenschluss aller Hersteller von PowerPC-Prozessoren und Komponenten für den PowerPC.

Der im PowerPC-Mac verwendete **PowerPC**-Prozessor wurde Anfang der 90er Jahre von Apple, Motorola und IBM auf Basis des POWER, welchen IBM in Großrechnern verwendete, entwickelt. Der aus 11 Chips bestehende POWER wurde zu einem Chip zusammengefasst und um einen von Motorola entwickelten Bus ergänzt. Der »Core« bzw. »Core 2« in den Macs mit Intel-Architektur ist ein **x86**, er entstand in einer langen Evolution aus dem 8086 (siehe Seite 20, Geschichte).

Apple verwendet dieses Logo anstelle des offiziellen Intel-Logos.

Der PowerPC im Mac ist Teil einer großen Familie mit unterschiedlichsten Einsatzgebieten und von verschiedenen Herstellern. PowerPCs werden unter anderem auch in Embedded-Computern (wie der Motorsteuerung im Auto), in den aktuellen Spielekonsolen oder auch in Supercomputern (wo er die obersten Plätze der Liste der schnellsten 500 Computer dominiert) und in Mainframes verwendet. Der x86 dagegen ist praktisch ein reiner Personal-Computer-Prozessor. Für die anderen Einsatzgebiete hat Intel andere Prozessoren im Programm – die RISC-Prozessoren Itanium und XScale (wurde jedoch 2006 an die Firma Marvell abgestoßen). XScale basiert auf dem »ARM« (Advanced RISC Machine), dem Prozessor im iPod und iPhone.

► 28 RISC

Prozessortechnik – oder – wie funktioniert ein Prozessor?

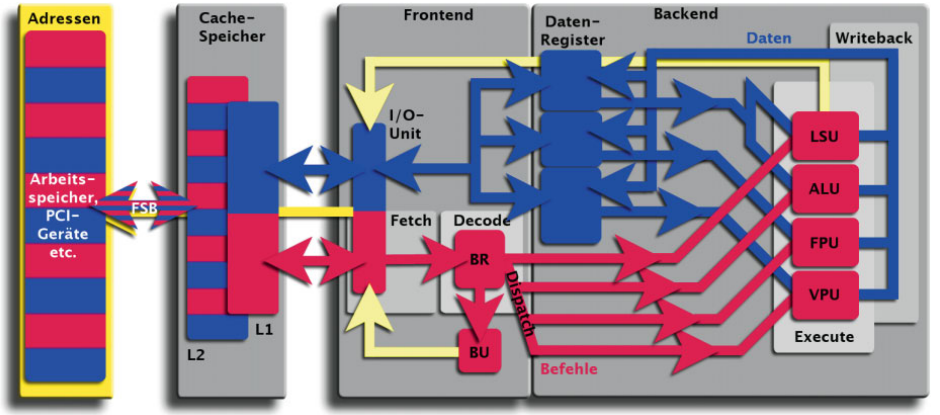
Ein Prozessor besteht aus verschiedenen funktionalen Einheiten, die erst das Abarbeiten von Programmen ermöglichen. Man kann hier zwischen dem Frontend und dem Backend unterschieden. Das Frontend ist dafür zuständig, dem Programmablauf zu folgen und das Backend mit den Befehlen und Daten aus dem Arbeitsspeicher zu versorgen. Das Backend übernimmt die eigentliche Berechnung der Daten anhand der Befehle.

Programmablauf

Alle Prozessoren arbeiten nach dem gleichen Ablauf – Fetch, Decode/Dispatch, Execute, Writeback: Ein Prozessorbefehl wird von einer bestimmten Speicheradresse angenommen, abgefertigt und verarbeitet (d.h. auf bestimmte Daten angewendet). Dann wird das Ergebnis zurückgeschrieben. Gleichzeitig wird der Programmzähler um Eins erhöht, und der nächste Befehl im Programmablauf in der nächsthöheren Speicheradresse wird angenommen. Diese Sequenz durchläuft der Prozessor immer und immer wieder, von dem Zeitpunkt, an dem der Computer eingeschaltet wird bis zum Ausschalten.

Register

Register sind die Speichereinheiten innerhalb des Prozessors, der einzige Speicher, auf den der Prozessor direkten Zugriff hat. Befehle können nur auf Daten angewendet werden, die sich in Registern befinden. Jeder Befehl und die zu bearbeitenden Daten müssen zuerst



Schematische Darstellung eines superscalaren RISC-Prozessors. In vier Pipeline-Stufen und mit vier parallelen Recheneinheiten werden die Daten (blau) mithilfe der Befehle (rot) verarbeitet. Die LSU berechnet die Adressen (gelb) für den Speicherzugriff.

BR = Befehls-Register
BU = Branch Unit
LSU = Load/Store Unit
ALU = Arithmetic Logical Unit
FPU = Floating Point Unit
VPU = Vector Processing Unit

in ein Register geladen werden. Um Daten aus dem Arbeitsspeicher in die Register zu laden, werden Lade-Befehle verwendet, bzw. Speicher-Befehle, um Daten aus Registern in den Arbeitsspeicher zu schreiben (Load/Store-Befehle). Die unterschiedlichen Recheneinheiten für Festkomma-, Fließkomma- und Vektor-Berechnungen besitzen jeweils eigenen Satz an Registern. Speicheradressen werden in den Festkomma-Registern gespeichert. Zusätzlich gibt es Befehlsregister, sowie Status- und Steuer-Register (z.B. für den Programmzähler).

Programmverzweigungen

Normalerweise wird ein Befehl eines Programmes nach dem anderen abgearbeitet. In bestimmten Situationen kommt es jedoch zu Sprüngen oder Schleifen. Diese werden von der **Branch-Unit** bearbeitet. Ein Branch-Befehl setzt den Programmzähler unter bestimmten Umständen auf einen anderen Wert, sodass die folgenden Befehle von einer anderen Speicheradresse geladen werden.

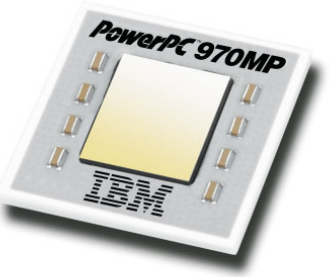
Aufbau von Prozessor-Befehlen

Ein Befehl besteht aus einer Reihe von Informationen. Die erste Information ist die Kennung, der »Opcode«. Dieser bestimmt, welche Berechnung ausgeführt werden soll (z.B. Addiere, Lade von Adresse etc.). Dahinter finden sich Registeradressen für die Quelle und das Ziel der Berechnung. Einige Befehle beinhalten auch eine so genannte »Immediate«, eine Zahl, die beim Übersetzen des Programms schon bekannt war.

Techniken in der Mikroarchitektur

- **Pipelines:** Bei den allerersten Prozessoren wurde ein neuer Befehl erst angenommen, wenn der vorherige Befehl fertig bearbeitet war. Damit verbringt aber ein Teil des Prozessors einen Großteil der Zeit mit Leerlauf. Daher ist man auf die Idee der Befehlspipeline gekommen. Der nächste Befehl wird schon angenommen, wenn der erste sich noch in Bearbeitung befindet. Damit sind immer alle Funktionseinheiten ausgelastet. Die kleinste Pipeline besteht aus vier Stufen (die klassische RISC-Pipeline – für die vier Phasen der Bearbeitung eines Befehls: Fetch, Decode/Dispatch, Execute, Writeback). Dieser Ablauf kann wiederum in kleinere Einheiten unterteilt werden. (Der erste G4 hatte ebendiese 4 Stufen, die später auf 7 erweitert wurden.

PowerPC-74xx-Prozessor (G4)



PowerPC-970-Prozessor (G5)

Abbildungen: Freescale, IBM

Der G5- und die Intel-Core-Prozessoren haben 12 bzw. 16 Pipeline-Stufen. Die längste Pipeline besitzt Intels Pentium 4. Dort sind es bis zu 31 Stufen.) Längere Pipelines ermöglichen eine höhere Taktfrequenz, da die kleineren Bearbeitungsschritte weniger Zeit benötigen.

Allerdings funktioniert das nur gut, solange es keine Abhängigkeiten von den Ergebnissen vorheriger Befehle oder Verzweigungen gibt. Dann müssen schon in Bearbeitung befindliche Befehle wieder gelöscht werden, es entstehen Blasen in der Pipeline, so genannte »Bubbles«. Je länger die Pipeline, desto öfter treten diese auf und größer sind sie.

- **Sprungvorhersage:** Bei Sprung-Befehlen ist es aber möglich, um Verzögerungen durch Bubbles zu vermeiden, bestimmte Befehle schon auf Verdacht auszuführen. Beispielsweise ist die Wahrscheinlichkeit, dass sich eine Schleife wiederholt ungleich höher, als dass die Schleife verlassen wird (z.B. wird beim Umwandeln eines Bildes mit 1000 Pixeln die Schleife 999 mal wiederholt, aber nur einmal verlassen). Moderne Prozessoren haben eine eigene Recheneinheit, die darauf spezialisiert ist, zu berechnen, welcher Zweig am wahrscheinlichsten als nächstes genommen wird (den Branch-Predictor).
- **Superscalare Prozessoren:** Die ersten Prozessoren besaßen nur eine Recheneinheit im Backend, mit der jede Art von Daten berechnet wurden. Später, als die Zahl der Transistoren zunahm, wurden in den Prozessor zuerst zwei gleiche Recheneinheiten eingebaut, die die Daten parallel berechnen konnten. Noch später wurden die Recheneinheiten in Unter-Einheiten für unterschiedliche Arten von Berechnungen geteilt (z.B. eine Einheit für die einfachen Berechnungen Addition, Subtraktion und Multiplikation, plus eine für die komplizierte Division). Beim PowerPC 970 und Intel Core arbeiten 12 bzw. 13 unterschiedliche Recheneinheiten parallel. Moderne Prozessoren arbeiten »superscalar«. Damit kann der Prozessor bei einem Takt nicht nur einen Befehl, sondern mehrere ausführen. Ein superscalarer Prozessor braucht jedoch eine zusätzliche Steuereinheit, die dafür sorgt, dass die Ergebnisse in der richtigen Reihenfolge ausgegeben werden.
- **Weitere Techniken:** Im Laufe der weiteren Entwicklung wurden weitere Techniken erfunden, die die Leistungsfähigkeit des Prozessors steigern sollten, wie das Umsortieren der Befehle (Out of Order Execution, OoOE) vor dem Ausführen oder das Ausführen von Befehlen auf Verdacht (Speculative Execution), unsichtbare Register, in denen

Daten zwischengespeichert werden, wenn ein Register belegt ist (Rename-Register) o.ä. Allerdings kommen einige moderne Prozessordesigns (wie die PowerPC-basierten Prozessoren in der XBox 360 und der Playstation 3) von manchen dieser Techniken wieder ab, da man – wie in den 80er Jahren bei CISC – gerade feststellt, dass sie u.U. mehr Kosten, als Nutzen bringen.

Speicher

Wie oben beschrieben hat der Prozessor nur auf die Register einen direkten Zugriff. Zugriff auf den Arbeitsspeicher ist nur über den Speicherbus anhand von Adressen möglich. Auch Ein- und Ausgabegeräte (die Festplatte, die Tastatur, die Grafikkarte o.ä.) werden über Adressen behandelt. In Programmen finden sich jedoch keine absoluten Speicheradressen (diese ändern sich, z.B. ja nachdem wie groß der eingebaute Arbeitsspeicher ist). Für das Umrechnen der in den Programmen gespeicherten Adressen in absolute Adressen besitzen heutige Prozessoren eine eigene Funktionseinheit, die **Load/Store-Unit**.

Cache

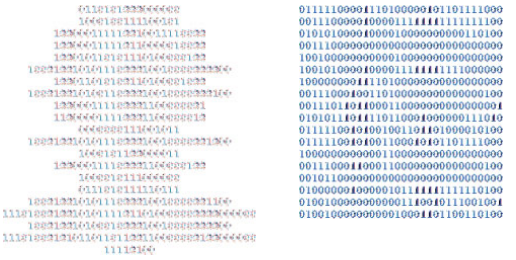
Regulärer Arbeitsspeicher ist um ein Vielfaches langsamer, als der Prozessor. Es gibt zwar schnelleren Speicher, doch der ist erheblich teurer (je schneller, desto teurer). Damit der Prozessor so wenig, wie möglich auf den langsamen Arbeitsspeicher warten muss, wird schneller Speicher in mehreren Stufen als Cache für den langsamen Arbeitsspeicher verwendet (wobei der Arbeitsspeicher wiederum ein Cache für die Festplatte ist). Im Cache werden Kopien der aktuell benötigten Daten und Programmbefehle zwischengespeichert. Es gibt folgende Cachestufen: Am schnellsten, aber auch am kleinsten ist der Level-1-Cache (L1), er ist in Prozessor integriert. Im L1 werden Befehle und Daten in getrennten Bereichen gespeichert. Auch der größere, etwas langsamere L2 befindet sich bei heutigen Prozessoren zusammen mit dem Prozessor auf einem Chip. Bei manchen Prozessoren (im Mac bei bestimmten Versionen des G4) steht zusätzlich noch ein Level-3-Cache zu Verfügung. Die Verwaltung des L1- und L2- (und evtl. L3-) -Cache übernimmt der Prozessor, sie ist von Prozessor zu Prozessor unterschiedlich. Mithilfe der Cache-Hierarchie wird der Speicherzugriff stark beschleunigt. Wie stark wird nicht nur durch die Größe des Caches, sondern auch durch seine Organisation bestimmt.

48 ▸ Software, Speicherverwaltung

Befehlssätze

Jede Prozessorfamilie arbeitet mit einem eigenen Satz von Prozessorbefehlen (Instruction Set Architecture, ISA). Die **ISA** wurde erstmals in den 60er-Jahren im IBM System/360 eingeführt, um die Programmierung von der eigentlichen Hardware unabhängig zu machen. Die ISA liegt als Abstraktionslayer über der Mikroarchitektur des Prozessors. Vorher musste ein Programm immer für die Mikroarchitektur angepasst werden, selbst bei einer kleinen Änderung des Prozessors mussten die Programme neu geschrieben werden. Mit Einführung der ISA konnte vorhandenen Software auch auf neuer Hardware mit der gleichen ISA verwendet werden. Die Befehle der ISA wurden dabei zuerst mithilfe der Mikrocode Engine verarbeitet. Diese ist eine Art Prozessor im Prozessor, der aus jedem Befehl eine Reihe von Befehlen erzeugt und diese dann dem Backend zugeführt.

Mit der Lösung des einen Problems wurde jedoch ein neues geschaffen: Der Markterfolg eines Prozessors wurde von der vorhandenen Software abhängig. So konnte der x86, obwohl seine ISA mit modernen Prozessortechniken absolut inkompatibel ist, sich trotzdem weiterhin



x86- und PowerPC-Code. Im x86-Code (rot) haben Befehle eine unterschiedliche Länge. Beim PowerPC (blau) sind alle Befehle gleich lang. Sie haben auch die gleiche Struktur. Damit der Intel-Prozessor die Befehle effektiv verarbeiten kann, muss er den x86-Code intern in Code umwandeln, der so ähnlich aussieht, wie der PowerPC-Code. (Die Bilder zeigen echten Maschinencode. Dieses Code-Segment wird beim Starten eines Mac-OS-X-Programms aufgerufen.)

am Markt halten und sich gegen die modernen Prozessoren im RISC-Design behaupten.

CISC

Die Möglichkeiten, die sich durch die ISA ergaben wurden in den 70er-Jahren auf die Spitze getrieben. Grund dafür war u.a. Speicher zu sparen – Speicher war rar und teuer und Prozessorlogik billiger zu produzieren, als Speicher. Außerdem sollte dem Programmierer, der damals noch auf Programmieren in der – sich mehr an der Maschine, als am Menschen orientierenden – Sprache »Assembler« angewiesen war, die Arbeit erleichtert werden. Daher wurde mehrere Befehle zusammengefasst und Befehle eingeführt, mit denen Speicheradressen direkt manipuliert werden können. Die Befehle einer solchen CISC-ISA (Complex Instruction Set Computing) können vom Prozessor nicht direkt verarbeitet werden.

Damit wurden die Prozessoren immer komplexer. Da die Ressourcen beschränkt waren, ist die Zahl der Register stark beschränkt. Heutzutage ist die Einsparung an Speicherplatz irrelevant, bei einem Mac-OS-X-Programm beträgt sie, wenn überhaupt, nur wenige Prozent. Außerdem wird heutzutage nicht mehr in Assembler programmiert, sondern in Hochsprachen, wie »C« oder »C++« bzw. »Objective C« für Cocoa. Beim Erzeugen eines Programms sorgt der Compiler dafür, die Hochsprache in die Prozessorsprache umzusetzten.

► 42 Cocoa

RISC

Mit der RISC-Technologie (Reduced Instruction Set Computing) – die in den 80er-Jahren entworfen wurde, um die Grenzen, die durch CISC gesetzt wurden, zu überwinden – besann man sich auf die Einfachheit der Prozessoren aus der Zeit vor CISC. Das Konzept der ISA wurde beibehalten, die ISA aber so angelegt, dass die Befehle direkt vom Prozessor verarbeitet werden können. Der Prozessor kommt ohne eine Microcode Engine aus, die verfügbaren Transistoren konnten statt dessen für die eigentlichen Recheneinheiten verwendet werden. RISC sollte eine höhere Taktgeschwindigkeit, aber auch höhere Rechengeschwindigkeit pro Takt ermöglichen. Techniken, wie Pipelines und superscalare Prozessoren wurden erst durch RISC möglich. Bei RISC gibt es, bis auf die speziellen Load- und Store-Befehle, keinen Zugriff auf Speicheradressen. Die Zahl der Register ist ungleich größer, als beim CISC. Aus der gleichen Länge und Struktur der Befehle (bzw. der sich daraus ergebenden fixen Länge des Opcodes) ergibt sich aber auch eine beschränkte Anzahl von Befehlen. Für den Assembler-Programmierer bedeutet RISC einen größeren Programmieraufwand.

Eigenschaften des x86 Befehlssatzes

Intels x86-Architektur ist ein Lehrbuchbeispiel für einen »komplexen Befehlssatz« CISC. Die Befehle haben eine extrem unterschiedliche

Länge (zwischen einem und 17 Bytes; im Durchschnitt 3,5, in tatsächlich verwendetem Programm-Code kann der Wert höher oder niedriger sein). Häufig gebrauchte Befehle sind kurz, um Speicher zu sparen. Da die Zahl der kurzen Befehle rein mathematisch beschränkt ist, gibt es – um möglichst viele Befehle zu ermöglichen – zusätzlich seltener gebrauchte, längere Befehle. Zum Teil handelt es sich um zwei Versionen des gleichen Befehls: Ein kurzer Befehl, der automatisch auf ein bestimmtes Register angewendet wird und ein langer, der auf jedes beliebige Register angewendet werden kann. Der x86-Befehlssatz enthält auch Befehle, die Speicherinhalte direkt berechnen (der Prozessor muss die Inhalte natürlich trotzdem in die Register laden, um sie berechnen zu können). Pro Funktionseinheit stehen 8 Register im 32-bit-Modus zur Verfügung bzw. 16 im 64-bit-Modus. Beim x86 muss der Programmierer also immer darauf achten, dass die Register nicht überlaufen. Der x86 kennt nur Befehle mit zwei Operanden, der Inhalt eines Registers (oder einer Speicheradresse) wird mit dem Inhalt eines anderen verändert (nach dem Schema A + B → A). Der x86 bearbeitet Daten »Little Endian«. D.h. wenn das Wort »UNIX« ein 64-bit-Wert wäre, würde es in den Registern im 32-bit-Modus als »NU« und »XI« bzw. im 64-bit-Modus als »XINU« gespeichert.

Abbildung: Intel

Intel Core-2-Duo-Prozessor

Heutige x86-Prozessoren

Bei heutigen x86-Prozessoren werden die x86-CISC-Befehle im Prozessor zu Mikrobefehlen umgewandelt und dann abgearbeitet. Diese Mikrobefehle besitzen ähnliche Eigenschaften, wie RISC-Befehle (denn sie orientieren sich wie diese an den Arbeitsabläufen eines modernen Prozessors). Auf die Mikrobefehle ist jedoch weiterhin kein direkter Zugriff möglich, sie sind auch unterschiedlich je nach x86-Prozessortyp. Dabei gibt es bei heutigen x86-Prozessoren mehrere Einheiten zur Umwandlung der CISC-Befehle in die internen Befehle. Einfache Befehle werden von mehreren parallel arbeitenden einfachen Umwandlungseinheiten in ein oder zwei interne Befehle umgewandelt, die komplexeren Befehle werden über ein Mikroprogramm in eine Sequenz von – teils duzenden – internen Befehlen umgerechnet. Heutige x86-Prozessoren nehmen die Befehle Paketweise an. Bevor diese weiter verarbeitet werden können muss der Prozessor immer zuerst anhand des Inhalts feststellen, ob es sich dabei um einen oder mehrere Befehle handelt.

Durch exzessive Anwendung moderner Prozessor-Techniken Techniken nach dem Umwandeln der x86-Befehle in Mikro-Befehle, wurde der CISC-Nachteil bei den heutigen x86ern zum Teil wieder wettgemacht. Für diese Umwandlung werden jedoch weiterhin viele Ressourcen verschwendet. Ein Pentium Pro, aus dem die aktuellen Core-Prozessoren entwickelt wurden, brauchte 40% seiner Transistoren, um die mit der modernen Prozessortechnik inkompatiblen x86-Befehle zu übersetzen. Da die verfügbaren Ressourcen sich mit jeder Prozessor-Generation vervielfachen, dachte man, man könnte diesen Overhead bald vernachlässigen. Heute aber wird mehr auf Stromsparen geach-

tet und Leistungssteigerungen bei Prozessoren werden nicht mehr durch Vergrößern eines einzelnen Prozessors und höhere Taktung erreicht, sondern durch Mehrprozessor-Technik. Mit der Vervielfachung kompletter Prozessorkerne steigt die Menge der von der x86-Übersetzungshardware verbrauchten Ressourcen nun doch wieder an.

Eigenschaften des PowerPC

Der PowerPC-Prozessor benutzt im Gegensatz zum x86 einen »reduzierten Befehlssatz« RISC. Im PowerPC-Befehlssatz sind alle Befehle vier Bytes lang und gleich strukturiert. Z.B. ist der Opcode immer 6bit lang. Daher kann der Prozessor die Befehle wie einen gleichmäßigen Datenstrom annehmen. Ein PowerPC besitzt 32 universelle Register pro Recheneinheit. Beim PowerPC haben die meisten Befehle drei Operanden (A + B → D), einige, speziell Vektor-Befehle, auch vier Operanden (A + B + C → D). Wie bei RISC üblich muss immer ein Zielregister (D) angegeben werden (welches aber auch identisch mit einem der Quellregister sein kann).

◀ 21 Geschichte, x86, Erweiterungen

Recheneinheiten

Ein Prozessor besteht nicht nur aus einer Recheneinheit, sondern aus verschiedenen, die für unterschiedliche Rechen- und Steueraufgaben zuständig sind. Frühere Prozessoren besaßen als Recheneinheit nur die Einheit für Ganzzahl-Berechnungen (ALU, Arithmetic Logical Unit). Seit längerem enthalten Prozessoren mit der **FPU** zusätzlich eine Einheit für Fließkomma-Berechnungen in doppelter Genauigkeit (d.h. für 64bit-Fließkomma-Werte). Alle Einheiten können auch mehrfach vorhanden sein.

◀ 21 Geschichte, x87

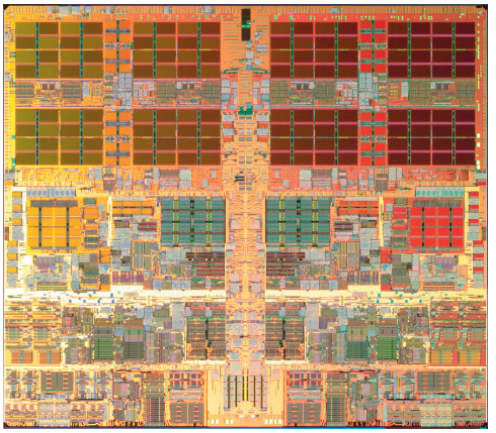
Vektor-Einheiten

Heutige Prozessoren enthalten zusätzlich zu den bekannten Recheneinheiten für Ganzzahl- und Fließkommaberechnungen eine weitere Recheneinheit, mit der multiple Daten bzw. Vektoren gleichzeitig mit einem Befehl bearbeitet werden können (SIMD – Single Instruction, Multiple Data).

- **AltiVec:** Die AltiVec- bzw. VMX-Einheit beim G4 bzw. G5– von Apple auch Velocity Engine genannt – ist 128 bit breit. Damit kann beispielsweise ein Befehl für 16 Datenworte à 8 bit Länge oder auch 4 Datenworte à 32 bit auf einmal ausgeführt werden. Wenn die zu verarbeitenden Daten es zulassen, macht Altivec den G4 bzw. den G5 zu einem der schnellsten Prozessoren überhaupt. Bei der Berechnung von rc5-Schlüsseln für <distributed.net> berechnet ein G4 mehr als die doppelte Anzahl an Schlüsseln wie ein doppelt so schnell getakteter x86-Prozessor. Da Altivec im G4 besser umgesetzt ist, übertrifft der hier sogar den schneller getakteten G5.
- **SSE** erfüllt beim Intel-Prozessor teilweise ähnliche Aufgaben wie AltiVec beim G4. Zusätzlich führt SSE auch normale 32- und 64-bit-Fließkomma-Befehle aus, ähnlich wie die Fließkomma-Einheit, die in jedem PowerPC vorhanden ist. Mit SSE können jedoch viele Rechenoperationen bei Weitem nicht so stark beschleunigt werden, wie mit AltiVec beim PowerPC.

Multicore

Die Prozessoren im letzten G5-PowerMac und in den Intel-Macs sind Dualcore-Prozessoren. Bei einem Dualcore-Prozessor sind zwei Prozessorkerne auf einen gemeinsamen Chip zusammengefasst. Wie bei einem Computer mit mehreren einzelnen Prozessoren kann dadurch die Rechenleistung gesteigert werden. Sie verdoppelt sich jedoch nicht, da zusätzlicher Verwaltungsaufwand entsteht. Mehrere Prozessorkerne auf einen Chip zu bringen hat gegenüber einzelnen Prozessoren den Vorteil, dass die Prozessoren direkt, ohne den Umweg



Blick ins Innere des PowerPC-970MP Dualcore-Prozessors. Hier kann man die beiden Cores und die unterschiedlichen Recheneinheiten erkennen. Der große Bereich ganz oben ist der L2-Cache-Speicher.

über die Northbridge, kommunizieren und auf einen gemeinsamen Cache-Speicher zugreifen können.

Der erste Dualcore-Prozessor wurde schon im Jahre 2001 eingeführt. Es war der POWER4 von IBM, aus dem später der PowerPC 970 (G5) – paradoxerweise zuerst als Singlecore-Prozessor – entwickelt wurde.

32 bit – 64 bit

Mit dem PowerMac G5 wurde mit dem PowerPC 970 der 64-bit-PowerPC-Prozessor am Mac eingeführt. Alle älteren PowerMacs verwenden 32-bit-Prozessoren. (Die Bezeichnung 64-bit-Prozessor bezieht sich nur auf die Breite der Register für die ALU.) Ein 64-bit-Prozessor bringt in erster Linie den Vorteil, dass mehr als 4 GB an Arbeitsspeicher adressiert werden können, es sind theoretisch 16 Exabyte möglich. Der **PowerPC** war schon von vorneherein als 64-bit-Prozessor geplant worden, die 32-bit-Version ist nur eine Untermenge. Der erste 64-bit-PowerPC-Prozessor wurde schon 1995 in der IBM AS/400 verbaut.

Aus diesem Grunde gibt es beim G5 keine Probleme mit der Kompatibilität. Alle alten 32-bit-Programme werden in voller Geschwindigkeit ausgeführt. Auch muss bei einem neuen 64-bit-Programm nicht jeder Programmteil aus 64-bit-Code bestehen. Im 64-bit-Programm-Modus können alle 32-bit-Befehle ausgeführt werden. Es können sogar in 32-bit-Programme einzelne 64-bit-Befehle eingefügt werden.

Die in den ersten Macs mit **Inte**l-Architektur verbauten Core-Duo-Prozessoren sind 32-bit-Prozessoren. Die Modelle mit Core-2-Prozessoren enthalten eine ursprünglich Anfang 2003 von AMD eingeführte 64-bit-Erweiterung – bei Intel EMT64 genannt. Über verschiedene Initialisierungsschritte wird der 32-bit-Prozessor zu einem 64-bit-Prozessor. Im 64-bit-Modus erscheint der Prozessor als ein anderer Prozessor, als im 32-bit-Modus, u.a. mit der doppelten Anzahl an Registern. 32-bit-Programme laufen dann in einem Kompatibilitätsmodus.

Laut Intel-Dokumentation kann die 64-bit-Erweiterung nur von einem 64-bit-Kernel mit 64-bit-Erweiterungen betrieben werden, 32-bit-Erweiterungen funktionieren dann nicht mehr. Unter Mac OS X jedoch wird der Prozessor mit einem 32-bit-Kernel und den vorhandenen 32-bit-Erweiterungen betrieben. Trotzdem laufen 64-bit-Programme. Apple hat ein paar Hacks in den x86-Kernel eingebaut, um dieses Problem zu umgehen (beim PowerPC ist kein Hack nötig, dort ist diese Möglichkeit so vorgesehen).

32 ▸ Northbridge

50 ▸ Software, 32-bit – 64-bit

Der Chipsatz

Auf dem Mainboard des Macs befinden sich neben dem Prozessor weitere Komponenten, die zum Betrieb eines vollwertigen Computers gebraucht werden. Bei diesen Komponenten spricht man vom »Chipsatz«.

Ein Chipsatz besteht meist aus zwei Hauptkomponenten. Die Northbridge kontrolliert den Speicher und verbindet den Prozessor mit dem lokalen Bus. Außerdem ist die Grafikhardware über die Northbridge angebunden. Die Southbridge ist für die Verwaltung der I/O-Schnittstellen, wie Festplatten, USB-, FireWire- und die Netzwerk-Schnittstellen zuständig. North- und Southbridge sind über den PCI-Bus bzw. über einen speziellen Bus verbunden.

Beim **PowerPC**-Mac wurden die Haupt- Chips für den Chipsatz von Apple selbst, auf den jeweiligen Mac zugeschnitten, entworfen – sie tragen Namen, nicht einfach nur Nummern.

► 35
PCIe-Lanes

► 35ff
Schnittstellen

In den G4-basierten Macs sind die FireWire- und die Gigabit-Ethernet-Schnittstelle direkt in die Northbridge integriert. Bei einigen Modellen sind die North- und Southbridge zu einen einzigen Chip – dem System-Controller – zusammengefasst. In der ersten Generation Uni-North und Keylargo zu Pangea, in einer späteren U2 und Keylargo zu Intrepid.

Die Macs mit **Intel**-Prozessor dagegen verwenden – wie die meisten PCs – auch einen Chipsatz von Intel. In den meisten Intel-Macs findet der 945-Chipsatz Verwendung. Der Chipsatz ist eigentlich für 32-bit-Prozessoren gedacht und kann daher – selbst, wenn mehr eingebaut ist – nur bis 3GB Speicher verwalten. (Beim dritten Modell des MacBook Pro bzw. iMac wurde auf den 965-Chipsatz gewechselt. Dieser kann mehr Speicher verwalten.) In der Variante 945GM, welche beim Mac mini und beim MacBook Verwendung findet, enthält der Chipsatz einen in die Northbridge integrierten Grafikprozessor (GMA 950). Dieser Grafikprozessor besitzt im Gegensatz zu normalen Grafikprozessoren keine eigene Speicherschnittstelle, er greift auf einen reservierten Teil des Hauptspeichers zu. Außerdem fehlen Recheneinheiten für bestimmte Grafikberechnungen, sodass der GMA diese Berechnungen an den Hauptprozessor zurück geben muss.

► 68
Systemstart

In die Southbridges ICH7 und ICH8 sind lediglich Festplatten- und USB-Schnittstellen integriert. Die Controller für die Netzwerk-Schnittstellen sind über PCIe angebunden, FireWire sogar nur über den langsamen PCI-Bus.

Im Mac Pro, in dem Xeon-Server-Prozessoren verbaut wurden, wird der Chipsatz 5000X verwendet. Dieser ist der einzige Chipsatz von Intel, der den Betrieb mehrerer Prozessoren erlaubt. Er stellt den PCIe-Steckplätzen deutlich weniger PCIe-Lanes zur Verfügung, als der im PowerMac G5 eingesetzte Chipsatz von Apple.

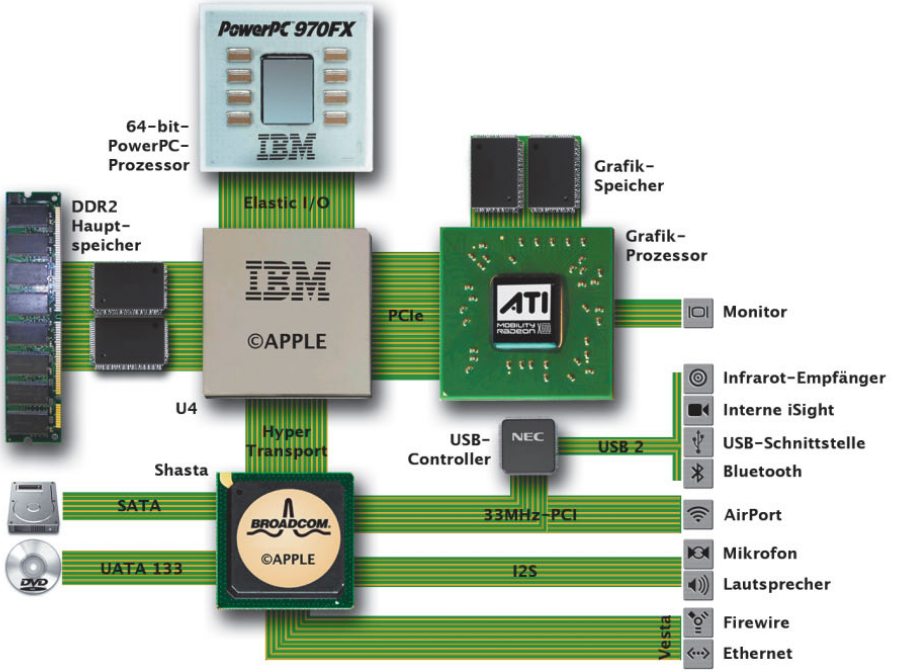
Firmware

Damit ein Computer starten kann, braucht das Mainboard ein eigenes kleines Betriebssystem, das das auf der Festplatte befindliche System starten kann. Die PowerPC-Macs verwenden hierfür die »Open Firmware«, die Intel-Macs verwenden »EFI«.

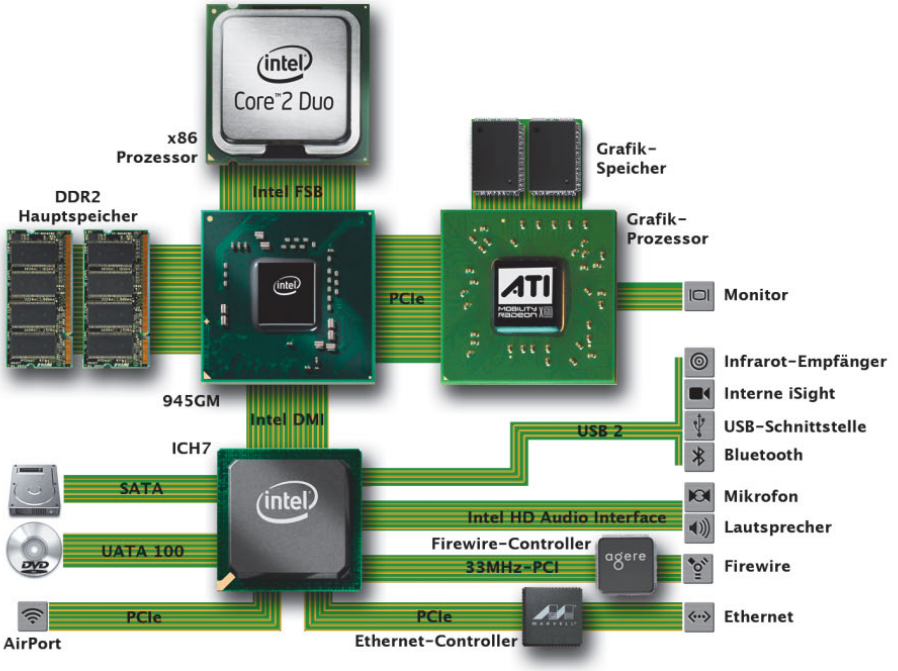
Open Firmware

Die Open Firmware ist ein von der Firma SUN entwickelter Industriestandard, der es ermöglicht, auf einem Computer ohne Probleme unterschiedliche Betriebssysteme verwenden zu können (z.B. Mac OS X, BSD-UNIX, Linux etc).

Die Open Firmware bildet anhand der angemeldeten Komponenten einen Device-Tree – eine Baumstruktur, in der alle Komponenten verzeichnet werden. Anhand dieses Device-Trees lädt der Mac-OS-X-Kernel die für die



Die wichtigsten Komponenten und ihre Verbindungen auf dem Board eines iMac G5 (iSight).



Das Design des Mainboards im Intel-iMac entspricht dem des MacBook Pro bzw. ohne den Grafikprozessor und -speicher dem des Mac mini und MacBooks – und dem Design der meisten PC-Notebooks mit Intel-Prozessor.

Komponenten auf dem Board und die Steckkarten notwendigen Treiber.

Die Open Firmware ist von Apple auf den jeweiligen Mac angepasst. Sie wird in einem speziellen Flash-Speicherchip gespeichert, mittels Updates kann Apple so eventuelle Fehler beheben oder Erweiterungen hinzufügen.

▸ 86 Praxis, BootCamp

Die Open Firmware bietet eine eigene Shell, eine Kommandozeilen-Oberfläche, mit der sie programmiert werden kann. Sie kann mit der Tastenkombination ⌘⇧OF beim Rechnerstart aufgerufen werden.

▸ 255 Praxis, Shell

EFI

In den Macs mit Intel-Prozessor übernimmt EFI (Extensible Firmware Interface) die gleichen Aufgaben, wie die OpenFirmware. Beim x86-PC ist dafür normalerweise das BIOS zuständig, das mittlerweile schon 25 Jahre alt ist. Es wurde zwar im Laufe der Zeit erweitert, setzt aber immer noch den »Realmode« des x86-Prozessors voraus (der Prozessor verhält sich wie ein 8086, mit entsprechend geringen Möglichkeiten bezüglich des adressierbaren Speichers und der Programmiersprache).

◂ 20 Geschichte, x86, BIOS

▸ 80 Praxis, Firewire-Targed-Modus

EFI wurde von Intel ursprünglich als Firmware für den 64-bit-Server-Prozessor Itanuim entwickelt. Beim x86 ist es eine Abstraktionsschicht und ein Interface-Aufsatz für das BIOS, das im Gegenzug auf das geringst mögliche Maß reduziert werden kann. EFI bietet für den Systemstart ähnliche Möglichkeiten, wie die Open Firmware – es ermöglicht das Auffinden der Startdatei auf der Festplatte und es können Firmware-Programme (hier »EFI-Module« genannt) ausgeführt werden. Den Zugriff auf die Kommandozeillen-Oberfläche ist jedoch bei Intel-Macs nicht ohne weiteres möglich, da Apple diese Möglichkeit gesperrt hat. Außerdem erstellt EFI keinen dynamischen Device-Tree, der die aktuelle Hardware widerspiegelt. Der Mac-OS-X-Kernel erstellt statt dessen einen »fakePPCDeviceTree«.

▸ 286 Praxis, NVRAM-Reset

▸ 261 Praxis, Flash-Speicher

Eines der Module, die beim Start ausgeführt werden können, liefert das BIOS-Interface. Dieses ist zum Start von Windows notwendig. (Bei den meisten PCs anderer Hersteller mit EFI ist dieses Compatibility Support Module (CSM) das einzige EFI-Programm, das ausgeführt wird. Es liefert dort auch die im PC-Bereich bekannte BIOS-Bedienoberfläche.)

EFI ist für den Betrieb von Mac OS X nicht notwendig, Mac OS X läuft mit wenigen Änderungen auf jedem beliebigen PC mit Intel- oder AMD-Prozessor. Allerdings versucht Apple dies zu verhindern.

Firmware-Programme

Open Firmware bzw. EFI können aber auch als eigenständiges kleines Betriebssystem fungieren. Im Firewire-Targed-Modus, wenn die Auswahl für das Boot-System aufgerufen wurde (beim Starten mit gedrückter ⌥-Taste) oder beim Hardware-Test mit der Hardware-Test-CD stellen Open Firmware bzw. EFI die benötigten Hardware-Treiber und die grafische Benutzeroberfläche.

Für den laufenden Betrieb könnte EFI dem Betriebssystem sogar eine komplette Treiberumgebung liefern, sodass das Betriebssystem keine eigenen Hardware-Treiber mehr besitzen muss. Mac OS X nutzt EFI aber nur für den Rechnerstart und benutzt im Betrieb die eigenen Treiber.

NVRAM

Die Einstellungen der Open Firmware bzw. des EFI werden im NVRAM gespeichert, einem nicht-flüchtigen (engl. non-volatile) Flash-Speicherbaustein auf dem Mainboard. Sie können auch im Terminal von Mac OS X mit dem Befehl »nvram -p« betrachtet werden.

PMU, SMU bzw. SMC

Macs haben auf dem Mainboard einen speziellen Baustein, der unter anderem die Batterie und die Lüfter überwacht. Bei den PowerBooks und iBooks ist das die PMU (Power Management Unit). Bei den G5-iMacs und -PowerMacs ist dies die SMU (System Management Unit).

Im Intel-Mac hat Apple nach Intel-Vorgabe den SMC (System Management Controller) verbaut, ein komplett vom Betriebssystem unabhängiges System aus einem kleinen RISC-Prozessor mit integriertem Arbeits- und Flash-Speicher, in dem ein eigenes kleines Betriebssystem gespeichert ist.

286 ▸
Praxis, PMU-, SMU- bzw. SMC-Reset

Interne Schnittstellen

PCI

Als interne Erweiterungssteckplätze stehen bei einigen Macs PCI-Steckplätze zur Verfügung. Es können 32- und 64-bit-PCI-Karten verwendet werden. Die Karten müssen dabei eine Mac-kompatible Firmware besitzen und es muss ein Treiber (Kernel-Extension) vorhanden sein, damit sie am Mac betrieben werden können. Einige Karten-Typen (z.B. Netzwerk-Karten) können jedoch auch in der normalen x86-PC-Version und ohne zusätzlichen Treiber betrieben werden.

PCI-Express-x1-Karte (1 Lane) mit zwei eSATA Schnittstellen

PCI-X

Der PowerMac G5 der ersten Generationen besitzt eine verbesserte Variante des PCI-Busses, PCI-X. Diese bietet deutlich höhere Geschwindigkeiten, ist aber auch zu 3,3V-PCI-Karten kompatibel. Ältere 5V-PCI-Karten passen nicht in den Steckplatz, kombinierte 3,3V-und- 5V-Karten werden mit 3,3V betrieben.

57 ▸
Kernel-Extension

AGP

Die Grafikkarte sitzt im AGP-Steckplatz, einer besonderen PCI-Variante, die einen höheren Datendurchsatz zulässt und der Grafikkarte die zusätzliche Möglichkeit gibt, direkt auf den Hauptspeicher zuzugreifen.

PCIe

Die dritte Generation des PowerMac G5 und der Mac Pro verwenden als interne Schnittstellen PCIe – auch PCI Express genannt. PCIe arbeitet mit Punkt-zu-Punkt-Verbindungen, so genannten Lanes. Karten bzw. Steckplätze können 1, 2, 4, 8 oder 16 Lanes besitzen. PCI-, PCI-X- und AGP-Karten sind nicht kompatibel zu PCIe.

▸ 264 Praxis, Festplatte installieren

▸ 262 Praxis, Bustypen

IDE

Die internen Festplatten, CD-Laufwerke etc. werden an die IDE-Schnittstelle angeschlossen. Die zwei internen Busse sind unterschiedlich schnell, ein langsamer (ATA33) für das CD-Laufwerk (und das ZIP-Laufwerk) und ein schneller (ATA66 oder ATA100) für die Festplatten. Es können zwei IDE-Geräte pro Bus angeschlossen werden. Eines muss als »Master«, das andere als »Slave« konfiguriert werden. Hängen zwei unterschiedlich schnelle Geräte an einem Bus, wird die Geschwindigkeit von dem langsameren Gerät bestimmt.

Serial ATA

Die Macs mit G5- und Intel-Architektur verwenden Serial ATA als Festplatten-Schnittstelle. Hier werden die Daten seriell übertragen. An einen Anschluss kann nur eine Festplatte angeschlossen werden, sie braucht nicht konfiguriert zu werden.

Serial-ATA-Festplatten können auch extern in speziellen eSATA-Gehäusen angeschlossen werden. Da Apple jedoch keine externen Anschlüsse einbaut, müsste dafür eine eSATA-Schnittstelle nachgerüstet werden. Das ist nur beim PowerMac oder Mac Pro mittels einer PCI- bzw. PCIe-Karte möglich.

USB-Anschlussstecker: A- (für Anschlussbuchse am Mac) und B-Form (bei einigen Peripheriegeräten verwendet)

Peripherie-Schnittstellen

Jeder Mac besitzt eine Reihe von Schnittstellen, über die er mit der Außenwelt kommuniziert. Einige dieser Schnittstellen sind bei jedem Mac vorhanden, andere nur bei bestimmten Modellen. Hier werden alle Schnittstellentypen und ihre Stecker beschrieben.

USB

Am Universal Serial Bus (USB) werden die Eingabegeräte – Tastatur, Maus – sowie andere »langsame« Geräte wie Modem, Drucker, Scanner usw. angeschlossen. USB ist ein serieller Bus, an dem die Geräte durch Hubs verbunden angeschlossen werden. Im laufenden Betrieb können bis zu 127 Geräte angeschlossen werden.

USB-Geräte können nicht direkt hintereinander angeschlossen werden. Sie müssen immer an einem Hub (den internen im Computer oder einen externen) angeschlossen werden. Die Geräte können nicht direkt miteinander kommunizieren. Die Kommunikation läuft immer über den Computer, selbst, wenn die Geräte an den gleichen Hub abgeschlossen sind.

USB 2

USB 2 ist eine stark beschleunigte Variante von USB. Hier können auch schnelle Geräte, wie Festplatten etc. angeschlossen werden. Hierfür sollten Sie aber FireWire den Vorzug geben. (USB2 wurde von Intel entwickelt um FireWire zu verdrängen. USB 2 bleibt aber trotzdem nur eine aufgebohrte Tastaturschnittstelle, ohne eine für eine Hochgeschwindigkeitsschnittstelle geeignete Bus-Topologie. Wegen der – um wenige Cent – billigeren Hardware werden häufig Geräte statt mit einer FireWire- mit einer USB-2-Schnittstelle ausgestattet.)

FireWire

In den meisten neueren Macs ist mit FireWire (IEEE 1394) eine Hochgeschwindigkeitsschnittstelle zu finden. FireWire ist ein serieller Bus, an den im laufenden Betrieb bis zu 63 Geräte angeschlossen werden können. FireWire kann beispielsweise zum Anschluss von externen Festplatten und digitalen Camcordern verwendet werden. Am Mac werden die 6-Poligen Stecker verwendet, die auch die Stromversorgung enthalten.

FireWire-Geräte sind meist mit zwei Schnittstellen – und einem sogenannten Repeater, der die Signale weiterleitet – ausgestattet, sodass mehrere Geräte hintereinander in einer Kette betrieben werden können. Im Gegensatz zu USB2 können die Geräte dabei auch direkt miteinander kommunizieren, der Computer muss die Kommunikation lediglich anstoßen.

FireWire 800

Mit FireWire 800 wurde die Geschwindigkeit der FireWire-Schnittstelle noch einmal verdoppelt. Dabei werden aber auch andere (9-Polige) Stecker verwendet. FireWire-400-Geräte können jedoch mit einem Adapter angeschlossen werden.

FireWire-Anschlussstecker:
FireWire 400 (4-Polig), FireWire 400 (6-Polig) und FireWire 800

PC-Card

PowerBooks sind mit einer PC-Card-Schnittstelle ausgestattet. In diese passen ein bis zwei PC-Cards vom Typ 2 oder eine PC-Card vom Typ 3. Die PC-Card-Schnittstelle bedient zwei Normen: **PCMCIA** und **Cardbus**. Das ältere, langsamere PCMCIA entspricht einer IDE-Schnittstelle, das neueren, schnellere Cardbus entspricht einer PCI-Schnittstelle.

ExpressCard/34

Das MacBook Pro besitzt statt der PC-Card-Schnittstelle eine ExpressCard-Schnittstelle. ExpressCard kombiniert eine PCIe-Schnittstelle mit einer USB-2-Schnittstelle in einem Steckplatz. Sie ist nicht kompatibel zu PC-Card. Im MacBook Pro können lediglich die schmaleren ExpressCard/34-Karten verwendet werden, nicht die breiteren ExpressCard/54-Karten.

Bluetooth

▸ 204
Praxis, Datenaustausch mit dem Handy über Bluetooth

Bluetooth ist ein Kurzstreckenfunk, der eigentlich für die Kommunikation zwischen einem Mobiltelefon und Komponenten konzipiert ist. Hier können aber auch andere Peripheriegeräte angeschlossen werden. Die neuesten Macs haben Bluetooth eingebaut, ältere können mit einem Adapter nachgerüstet werden, der an die USB-Schnittstelle gesteckt wird. Bluetooth bietet eine Übertragungsgeschwindigkeit von maximal 723 kbit, mit der neueren Version 2.0 wurde die Geschwindigkeit verdreifacht.

▸ 348
Referenz, Kontrollfeld Monitore

IrDA

Über den Infrarotanschluss können andere Computer oder Drucker drahtlos mit dem Mac verbunden werden. Dieser Anschluss ist bei einigen PowerBooks vorhanden. Die in den Intel-Macs verbaute IR-Schnittstelle für die »Apple Remote« Fernbedienung ist nicht mit IrDA kompatibel.

Monitor-Anschlussstecker: VGA und DVI-I (Singlelink)

Monitor

Monitore werden bei neueren Macs über die **DVI**-Schnittstelle angeschlossen. Hierbei wird die Konfiguration DVD-I verwendet, die neben den Pins für das digitale Monitorsignal auch Pins für analoges S**VGA**-Monitorsignal enthält. Für den Anschluss eines analogen Monitors liegt meist ein Adapter bei.

Einige Modelle besitzen eine **Mini-DVI**-Schnittstelle, an die DVI- oder VGA-Monitore mit einem Adapter angeschlossen werden können.

Der Apple Display Connector (**ADC**), welcher im PowerMac G4 verwendet wird, vereint das digitale Monitorsignal mit der Spannungsversorgung und einem USB-Anschluss in einem Stecker.

Für den Anschluss eines zweiten Monitors steht bei einigen Modellen eine zusätzliche DVI-Schnittstelle bzw. ein zusätzlicher Monitoranschluss nach der SVGA-Industrienorm – bei wenigen Modellen als Mini-VGA - zur Verfügung.

Monitore melden sich über DDC am Mac an; so erkennt Mac OS X automatisch den Monitortyp.

Video

Einige Macs haben als Zusatzausstattung einen Videoein- bzw. -ausgang. Für Composit-Video wird ein Chinch-Stecker verwendet, für S-Video ein MiniDIN8-Stecker.

Bei einigen Modellen kann die DVI-, Mini-DVI- oder Mini-VGA-Schnittstelle mittels Adapter als Videoausgang verwendet werden.

iSight

Der iMac G5 der zweiten Generation und die Intel-iMacs und -Notebooks sind mit einer integrierten Kamera ausgestattet, die bei Apple iSight genannt wird. Diese ist intern über USB angeschlossen. Ältere Geräte können mit einer externen iSight über FireWire ausgestattet werden. Alternativ können auch USB-Kameras anderer Hersteller verwendet werden. Für die iSight stellt Mac OS X verschiedene Software-Funktionen zur Verfügung.

Ton

Schon der erste Mac hatte eine integrierte »Soundkarte«. Alle neueren Macs können mit ihrer Soundausstattung CD-Qualität produzieren.

Alle Macs haben einen Line-Ausgang – manche Modelle auch zusätzlich einen Kopfhörerausgang und einen Line-Eingang – mit 3,5-mm-Miniklinken-Steckern. (Der Line-Eingang ist ein High-Level-Eingang, kein Mikrofoneingang. Die meisten Mikrofone brauchen hier einen Vorverstärker.)

Der PowerMac G5 verfügt außerdem jeweils über einen optischen digitalen Audioeingang und einen Audioausgang mit Toslink-Anschlüssen.

Apple Remote (Infrarot-Schnittstelle)

Der iMac G5 der letzten Generation und die Intel-Macs (außer Mac Pro) sind mit einer Infrarot-Schnittstelle für die Fernbedienung »Apple Remote« ausgestattet. Diese ist nicht mit IrDA kompatibel.

Als Batterie für die Fernbedienung wird eine CR2032-Knopfzelle verwendet.

Netzwerk-Schnittstellen

158ff ▸
Praxis, Netzwerk

350ff ▸
Referenz, Kontrollfeld Netzwerk

Ethernet

Schon seit vielen Jahren werden Macs mit einer Ethernet-Schnittstelle ausgerüstet. Für den Anschluss an das 10BaseT-, 100BaseT- bzw. 1000BaseT-Netz steht eine Buchse für RJ45-Stecker zur Verfügung.

AirPort

Macs haben mit der (je nach Modell optionalen oder serienmäßigen) AirPort-Karte die Möglichkeit, Netzwerkverbindungen über Funk herzustellen. AirPort benutzt das Ethernet-Protokoll.

372f ▸
Referenz, Kontrollfeld Ton

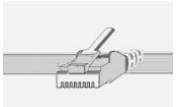
FireWire

Bei Bedarf kann auch die FireWire-Schnittstelle zur Übertragung von Ethernet-Paketen verwendet werden.

321 ▸
Referenz, Audio-MIDI-Konfiguration

Modem

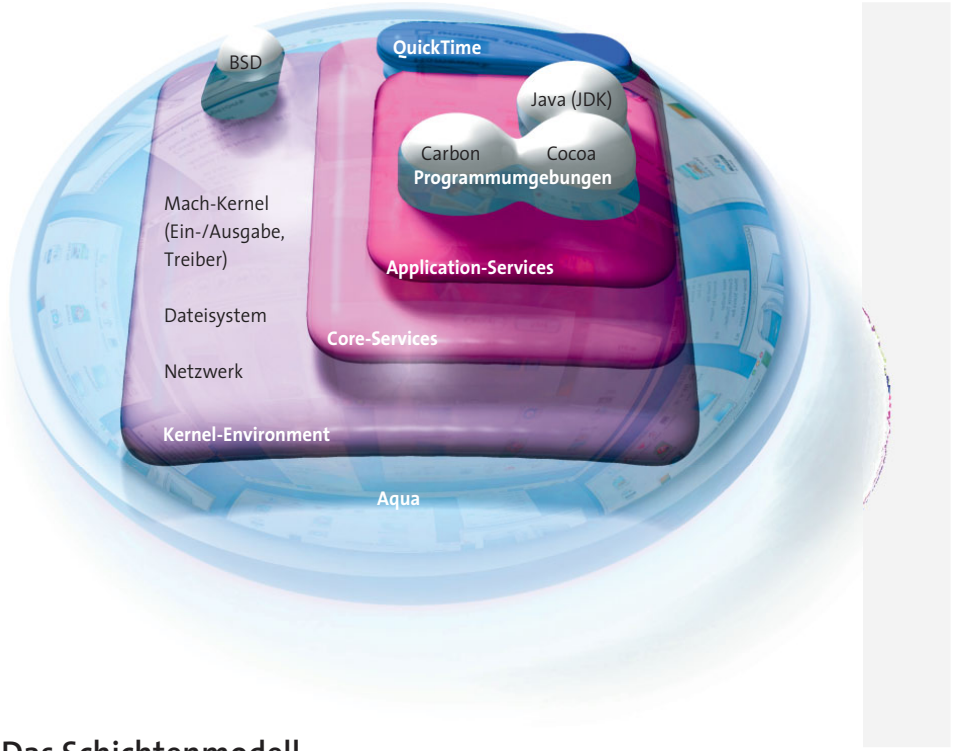
Die meisten PowerPC-Macs sind mit einem 56k-Modem ausgestattet. Dieses kann mit einem handelsüblichen Modemkabel mit RJ11-Stecker an das analoge Telefonnetz angeschlossen werden.



Ethernet-Stecker

Psychologie

Technisches zum Betriebssystem Mac OS X



Das Schichtenmodell

Mac OS X ist aus verschiedensten Wurzeln zusammengewachsen. Ein Teil ist klassisches UNIX, ein Teil kommt von NeXTStep, wieder ein anderer Teil aus dem Mac OS.

Mac OS X ist in Schichten aufgebaut, die verschiedene Aufgaben bei der Verarbeitung der Daten übernehmen. Dabei liegen die Schichten jedoch nicht wirklich aufeinander, eine Schicht kann mit jeder anderen kommunizieren.

Die unterste Schicht ist der **Mach-Kernel**, ein so genannter Micro-Kernel. Der Kern selbst enthält lediglich die Komponenten für Speicher und Prozessverwaltung. Treiber werden in Form von **Kernel-Extensions** (KEXT) dem Kernel hinzugefügt. Diese können dynamisch geladen und entladen werden. Der Kernel und die Extensions – zusammen auch XNU oder **Kernel-Environment** genannt – verhalten sich zusammen wie ein sogenannter monolithischer Kernel. Sie liegen beispielsweise in einem gemeinsamen Speicherbereich, was Vorteile in der Geschwindigkeit bringt. Nur das Kernel-

An ihrem Aussehen lassen sich die Programmumgebungen nicht unterscheiden. Von Links nach Rechts: Safari ist Cocoa, iTunes ist Carbon. Der Finder in Leopard ist ein Mix von beiden Umgebungen.

Environment darf direkt mit der Hardware kommunizieren. Alle anderen Prozesse laufen abgeschirmt.

Hier werden beispielsweise verschiedene Dateisysteme direkt oder über das Netzwerk angeschlossener Datenträger in ein gemeinsames virtuelles Dateisystem übersetzt, mit dem die Programme dann arbeiten können, ohne sich um die tatsächliche Struktur kümmern zu müssen.

Auf dem Kernel-Environment liegen die **Core-Services**, die den Programmumgebungen nicht-grafische Dienste anbieten. (Das Kernel-Environment zeigt sich zusammen mit einem Teil der Core-Services als Variante von BSD und ist von Apple unter dem Namen »Darwin« als Quellcode freigegeben worden. BSD – Berkeley Software Distribution – ist ein im Quellcode frei verfügbares UNIX-artiges Betriebssystem.) Über den Core-Services liegen die **Application-Services**. Diese stellen den Programmen die grafische Umgebung zur Verfügung.

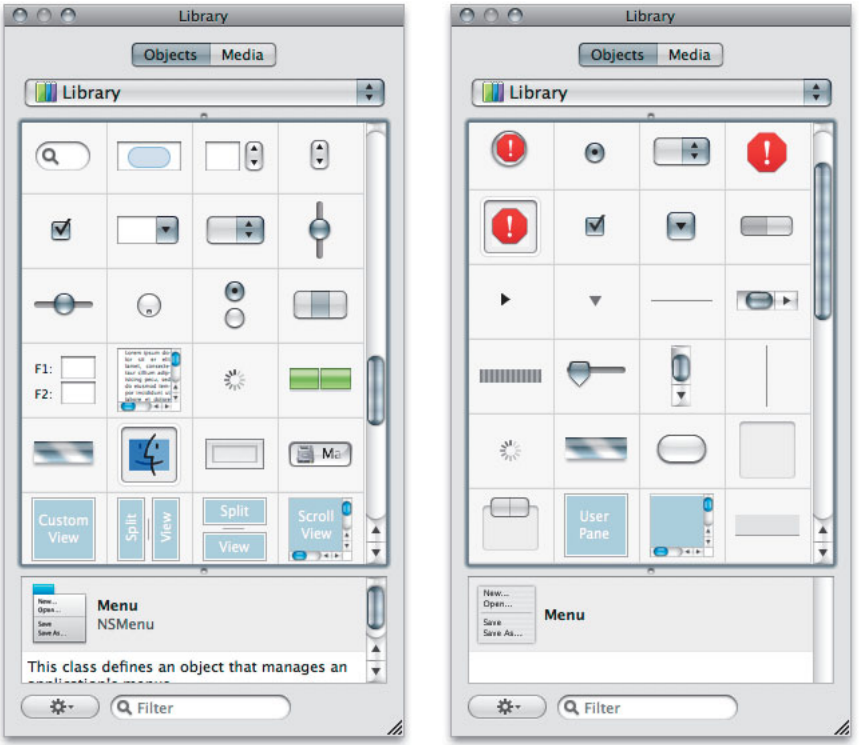
Die Programme selbst benutzen die verschiedenen, wiederum in einer übergeordneten Schicht liegenden **Programmumgebungen**. Jede Programmumgebung ist eine Sammlung so genannter »**API**s« (Application Programming Interfaces, Schnittstellen für Programme), auf die die Programme zugreifen können.

Wenn ein Programm eine bestimmte Funktion benötigt, die schon Teil des Systems ist, kann es diese aufrufen. Dank dieser Technik müssen viele wichtige Routinen nicht für jedes Programm neu programmiert werden. Das Programm muss lediglich eine Anfrage an den zuständigen Teil seiner Programmumgebung richten. So braucht ein Programm beispielsweise keine eigenen Routinen, die Fenster oder Schaltflächen erzeugen. Es muss nur die Funktionen erzeugen, die mit der Schaltfläche im Fenster aufgerufen werden.

Hexley, ein Schnabeltier, ist das Maskottchen von Darwin.

Programmumgebungen

Da Mac OS X ein aus unterschiedlichen Wurzeln zusammengewachsenes System ist, existieren hier verschiedene Programmumgebungen nebeneinander. Welche Umgebung ein Programm benutzt hängt davon ab, welche Programmiertechnik der Programmierer bevorzugt. Für den Anwender macht es praktisch keinen Unterschied, in welcher Programmumgebung ein Programm läuft. Er kann einfach zwischen den Programmen wechseln oder



Die Programmumgebungen Cocoa und Carbon bieten den Programmen annähernd die gleichen Möglichkeiten. Sie sind jedoch unterschiedlich organisiert. Diese Abbildungen stammen aus dem »Interface Builder«, einem Programm, mit dem die Benutzeroberflächen von Carbon- und Cocoa-Programmen zusammengestellt werden können. Es ist Teil der kostenlosen Entwicklersoftware auf der Developer-Tools-CD.

▸ 206f

Praxis, Daten zwischen Programmen austauschen

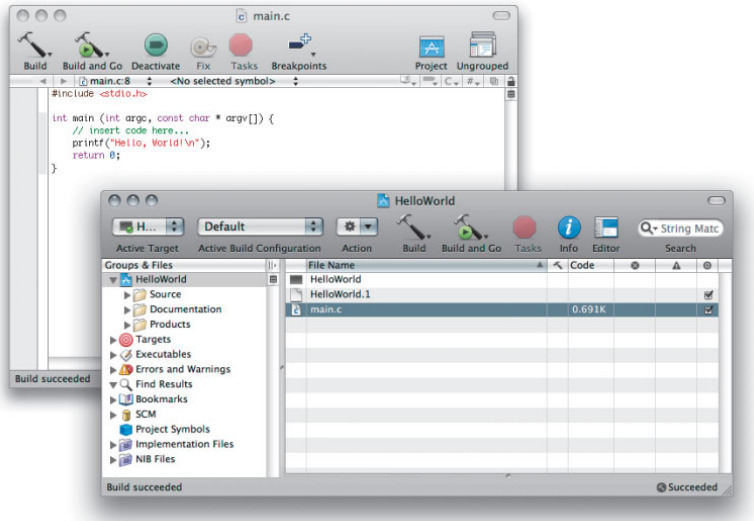
Daten zwischen ihnen austauschen. Der Funktionsumfang von Carbon und Cocoa ist außerdem immer in etwa auf dem gleichen Stand. Seit Mac OS X 10.2 können Cocoa-Programme auch Carbon-Interface-Elemente enthalten bzw. Carbon-Programme Cocoa-Elemente.

Cocoa

Cocoa ist die aus NeXTStep entstandene Programmumgebung. Sie verwendet die eigens mit NeXTStep zusammen entwickelte Programmiersprache Objective-C. Diese ist eine verhältnismäßig einfach zu programmierende und sehr leistungsfähige objektorientierte Programmiersprache, die jedoch nicht sehr verbreitet ist. Alternativ kann aber auch Java, das dann in XCode in Objective-C umgesetzt wird, und das neue Objective C++ als Programmiersprache verwendet werden.

Carbon

Carbon ist eine für das Mac OS X angepasste Version der Mac-OS-Programmumgebung. Ein großer Teil der APIs entspricht denen des



Apples XCode ist seit dem Wechsel zur Intel-Architektur die einzige Entwicklungsumgebung zum Erzeugen von Mac-OS-X-Programmen.

klassischen Mac OS. In Mac OS X werden diese APIs weiterentwickelt und es kommen viele neue hinzu. Für den Programmierer brachte Carbon den Vorteil, dass er sein Projekt auf vorhandenem Code aufbauen konnte. Außerdem konnte er seine aus dem Mac OS vertrauten Programmierwerkzeuge benutzen. Um Carbon-Programme für Intel zu erzeugen, müssen die Projekte jedoch in die XCode-Entwicklungsumgebung importiert und dort weitergeführt werden. Carbon ist die geeignetere Umgebung für die Portierung von Windows-Programmen.

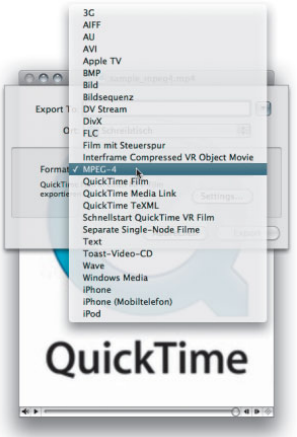
Carbon-Programme können auf unterschiedliche Weise programmiert werden. Reine Mac-OS-X-Carbon-Programme werden direkt gegen den Mach-Kernel verlinkt (Carbon-Mach-O). Nur diese Programme können die Möglichkeiten von Carbon unter Mac OS X voll nutzen.

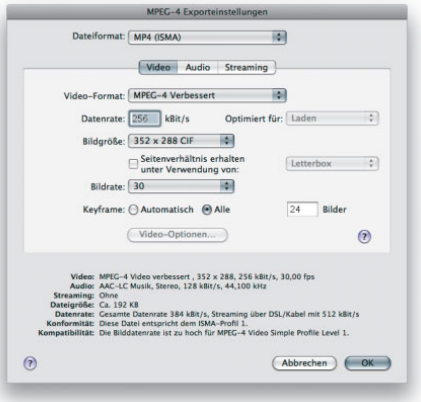
Carbon-Programme, die sowohl auf Mac OS X als auch mit der Systemerweiterung »CarbonLib« auf Mac OS 8 und 9 laufen können sollen, werden gegen eine weitere Zwischenschicht, den »Code Fragment Manager«, verlinkt (Carbon-CFM). Der Code Fragment Manager ist ein essentieller Bestandteil des klassischen Mac-OS-Systems und ist in Mac OS X im PowerPC-Teil des Carbon-Frameworks für die Kompatibilität enthalten. Diese Programme können nicht für den Intel-Prozessor angepasst werden, da das Design des Code Fragment Managers stark an die Möglichkeiten des PowerPC-Prozessors angelehnt war. Sie sind deshalb nicht als Universal-Binary erhältlich. Sie laufen auf einem Mac mit Intel-Prozessor nur unter Rosetta.

44 ▸
Rosetta

Java

Als weitere Programmumgebung steht eine Java-VM (Virtuelle Maschine) zur Verfügung. Java wurde von der Firma SUN mit dem Ziel entwickelt, dass Programmierer Programme schreiben können, die auf jeder beliebigen Plattform ausgeführt werden können. Solche in »100% pure Java« programmierten Programme





Der ausführbare Code des Programms »QTAmateur« ist als Universal Binary gerade mal 112kB groß. Trotzdem kann das Programm mit den unterschiedlichsten Videoformaten umgehen.

laufen also ohne jede Änderung auch unter Windows oder Linux. Sie werden im Quelltext geliefert und erst bei der Programmausführung für den entsprechenden Prozessor übersetzt.

▸ 255ff
Praxis, Shell

QuickTime

Für Multimedia-Inhalte bietet Apple sein bekanntes QuickTime. QuickTime kann mit unterschiedlichsten Arten von Multimedia-Daten umgehen. Diese werden dann in einem Container an die Programme weitergegeben. Das Programm selbst muss dafür nicht mit den Multimedia-Formaten umgehen. Der QuickTime-Container kann beispielsweise mehrere parallele Video- und Audiospuren enthalten, die in verschiedenen Formaten mit unterschiedlichen Codecs digitalisiert wurden. Im Programm wird lediglich ein Video abgespielt.

◂ 27
Befehlssätze

So können Programme, die die Dienste von QuickTime in Anspruch nehmen, mit einer neuen Version von QuickTime auch Dateiformate verarbeiten, mit denen sie vorher nicht umgehen konnten.

QuickTime gibt es außer für Mac OS X auch für Windows und – in einer älteren Version – für das klassische Mac OS.

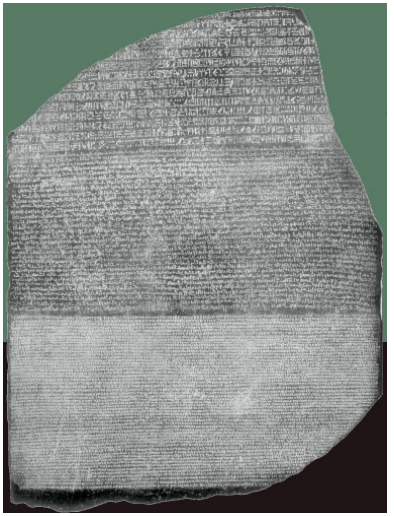
BSD-Kommandos

Das BSD-Subsystem greift direkt auf das Kernel-Environment zu. Es umgeht damit die Core-Services und die Grafik-Layer. Die UNIX-Shell kann über das Terminal-Programm bedient werden.

Auch die Daemons – unsichtbare Hintergrundprozesse, die Dienste bereitstellen – laufen in der BSD-Umgebung. (Die Namen dieser Prozesse enden idR. mit einem »d«, z.B. ist »diskarbitration**d**« ein Daemon, der für das automatische Anmelden von externen Festplatten sorgt. Das Wort »Daemon« leitet sich nicht vom mittelalterlich christlichen Dämonen ab sondern vom Altgriechischen bzw. von der sokratischen innerern Stimme »Daimonion«.)

Rosetta

Da die Intel-Prozessoren mit anderen Befehlen arbeiten als die PowerPC-Prozessoren, können diese eigentlich keinen Programmcode ausführen, der für PowerPC erstellt wurde. Um dieses Problem zu lösen, wurde in Mac OS X für Intel-Prozessoren Rosetta eingeführt. Rosetta übersetzt dafür den PowerPC-Code des Programms in x86-Befehle. Dieser Prozess passiert unsicht-



Der Emulationsmodus zum Ausführen von Mac-OS-X-Programmen für PowerPC auf Intel-Architektur wurde nach dem Stein von Rosetta benannt. Auf diesem Stein aus dem Jahre 196 vor Christus findet sich derselbe Text in griechisch, demotisch und ägyptisch eingemeißelt. Der Stein von Rosetta brachte den Durchbruch bei der Entschlüsselung der ägyptischen Hieroglyphen.

bar im Hintergrund, sodass sich Programme, die mittels Rosetta ausgeführt werden, nicht anders verhalten als andere Programme. Der Benutzer merkt keinen Unterschied.

Allerdings werden Programme unter Rosetta aufgrund der Übersetzung mit einer geringeren Geschwindigkeit ausgeführt als Programme, die für den Intel-Prozessor erstellt wurden.

Beim Wechsel vom 68k-Prozessor zum PowerPC war es durch die Möglichkeiten des PowerPC-Prozessors möglich, nur Teile eines Programmes auf den neuen Prozessor anzupassen. Das ist mit dem x86-Prozessor nicht möglich. Hier muss der Code eines Prozesses immer im Ganzen für eine Architektur vorliegen. Daher können Kernel-Extensions und Programme, die auf eine Kernel-Extension angewiesen sind, Plug-Ins und Erweiterungen für die Systemeinstellungen (PreferencePanes), sowie bestimmte Java-Applets nicht unter Rosetta ausgeführt werden.

Auch gibt es ein paar Einschränkungen dabei, welche Art von Code über Rosetta ausgeführt werden kann. G3- und G4-Programmcode wird ausgeführt, nicht jedoch Programmcode, der auf den G5-Prozessor angewiesen ist.

Rosetta übersetzt den Programmcode nach Bedarf in Häppchen. Beim Start des Programms wird der dabei verwendete Code übersetzt, optimiert und in einen Cache geschrieben. Wenn im Laufe des Programmbetriebs ein anderer Teil des Codes gebraucht wird, wird dieser zusätzlich übersetzt und in den Cache geschrieben.

X11

Optional kann in Mac OS X mit X11 eine weitere Programmumgebung installiert werden. X11 ist die Standard-Grafikumgebung für UNIX-Programme. Apples X11-Programm bildet dabei die X11-Grafiken in Aqua-Fenstern ab.

259 ▸
Praxis, X11

Classic

Die Classic-Programmumgebung, in der auf PowerPC-Macs Programme für das klassische Mac OS unter Mac OS X ausgeführt werden können, wird von Mac OS X Leopard nicht mehr unterstützt.

Unter Quartz Extreme werden alle Objekte der Benutzeroberfläche zu OpenGL-3D-Objekten. Diese werden von der Grafikkarte miteinander arrangiert.

Grafik

Mac OS X verwendet für die Darstellung der Objekte auf dem Bildschirm verschiedene Technologien. Je nach Art des Objekts und vom Programmierer bevorzugter Technologie kommen sie parallel zum Einsatz.

Quartz

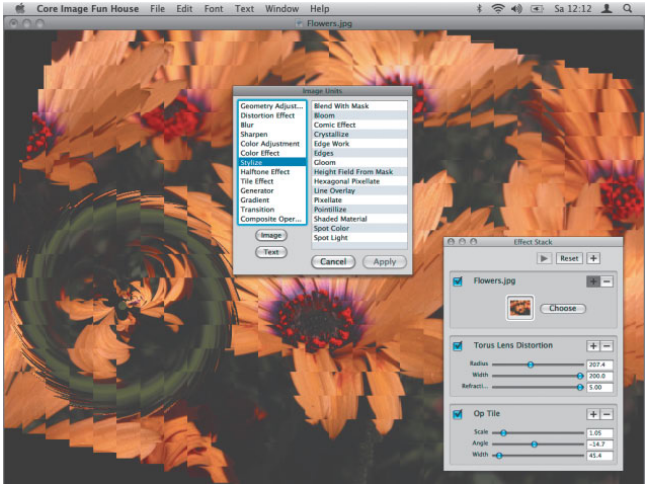
Quartz basiert auf dem von Adobe entwickelten PostScript-Format »PDF« (Portable Document Format), in dem Layouts unabhängig von der verwendeten Computerplattform (oder dem Drucker) immer gleich dargestellt werden. In PDF werden Objekte anhand von Vektoren und mathematischen Formeln beschrieben. Dadurch können alle Objekte beliebig skaliert werden. Sie werden in jeder Größe gleich scharf dargestellt.

QuickDraw

Zur Darstellung von zweidimensionalen Objekten in älteren Carbon-Programmen bietet Mac OS X eine erweiterte Version von **QuickDraw**, dem Grafiksystem des klassischen Mac OS. Dieses wird aber seit 10.4 nicht mehr weiterentwickelt.

Open GL

Zur Darstellung von dreidimensionalen Objekten verwendet Mac OS X **Open GL**. Open GL



Die kleine Beispielapplikation »Core Image Fun House« in den Developer Tools zeigt die Möglichkeiten von Core Image.

wurde von Silicon Graphics als plattformunabhängige Bibliothek zur Darstellung von 3D-Objekten entwickelt und ist mittlerweile der Industrie-Standard für 3D-Darstellung. Moderne Grafikkarten können Open-GL-Objekte ohne Hilfe des Hauptprozessors berechnen.

Viele Spiele und 3D-Programme für Windows oder UNIX verwenden Open GL. Diese lassen sich relativ einfach auf Mac OS X portieren, da die Routinen für die Darstellung der 3D-Objekte nicht verändert werden müssen. Im Unterschied zu Windows und Linux ist OpenGL in Mac OS X direkt in das System integriert und wird nicht einzig und alleine von dem Spiel bzw. 3D-Programm benutzt (siehe nächsten Absatz zu QuartzExtreme). Daher kann es vorkommen, dass – obwohl Mac OS X die modernste OpenGL-Implementation überhaupt besitzt – diese Spiele auf dem Mac weniger schnell laufen als auf einem vergleichbaren PC.

Quartz Extreme

Ab Mac OS X 10.2 die Grafik mit Quartz Extreme aus. Quartz Extreme verwendet OpenGL zur Grafikausgabe. Die Objekte (z.B. Fenster) werden als 3D-Objekte, die Inhalte der Objekte als Texturen (Oberflächen) der Grafikkarte zugeführt. Sobald jetzt ein Objekt bewegt wird, kann die Grafikkarte mit ihrem eigenen Prozessor die Objekte neu anordnen, der Hauptprozessor wird von dieser Rechenarbeit entlastet.

Quartz Extreme benötigt ATI-Radeon- oder NVidia-Geforce-Grafikkarten mit mindestens 16 MB Grafikspeicher. Es wird bei älteren Grafikkarten – wie der ATI Rage128, die in vielen PowerMacs verbaut wurde – nicht aktiviert. Älteren Grafikkarten fehlen wichtige Funktionen im Grafikprozessor, wie bestimmte Transparenzeffekte etc. Ein einfaches Beispiel sind unterschiedlich große Texturen: Da jedes Fenster unterschiedlich groß ist, ist auch der Fensterinhalt und damit die Textur für das Fenster unterschiedlich groß. Die ältere Grafikkarte kann jedoch nicht mit unterschiedlich großen Texturen umgehen, also bleibt Quartz Extreme abgeschaltet und das normale Quartz wird benutzt.

Auch der im Intel-Mac-mini und im MacBook eingesetzte Intel-Grafikprozessor GMA 950 beherrscht nicht alle Funktionen moderner

◂ 27
Hardware, Speicher

Grafikhardware. Er suggeriert jedoch dem System, dass er die Funktionen beherrsche und gibt einen Teil der Berechnungen dann wieder an den Hauptprozessor ab.

In Mac OS X 10.5 werden – wenn eine geeignete Grafikkarte mit 64 MB Speicher oder mehr vorhanden ist – zusätzlich auch die Fensterinhalte auf der Grafikkarte berechnet (Quartz 2D Extreme).

QuickTime

Zur Darstellung von Multimedia-Inhalten wie Videos etc. dient QuickTime. QuickTime arbeitet jedoch nicht eigenständig, sondern braucht in jedem Fall zum Darstellen der Multimedia-Inhalte eine Host-Application in Form eines Anwendungsprogramms bzw. eines Browsers.

▸ 316
Praxis, QuickTime Player

CoreImage, CoreAudio, CoreVideo und CoreAnimation

Mit CoreImage, CoreAudio und CoreVideo hat Apple Funktionen für Bildbearbeitungs-, Audio- und Video-Bearbeitungsprogramme in das Betriebssystem integriert. Core Animation ermöglicht Animationseffekte in der grafischen Oberfläche. Entsprechend programmierte Programme können, statt eigene Filter zu benutzen, die in das Betriebssystem integrierten Units verwenden. Das besondere an den Units von CoreImage, CoreVideo und CoreAnimation ist, dass die Filterfunktionen direkt in der Grafikkarte ausgeführt werden. (Das ist jedoch nur möglich, wenn die Grafikkarte einen dafür geeigneten Grafikprozessor besitzt.) Damit werden Anwendungen von Filtern, der Berechnung selbst auf einen G5 einige Zeit benötigt, in Echtzeit möglich. Gleichzeitig bleibt der Prozessor für andere Aufgaben verfügbar.

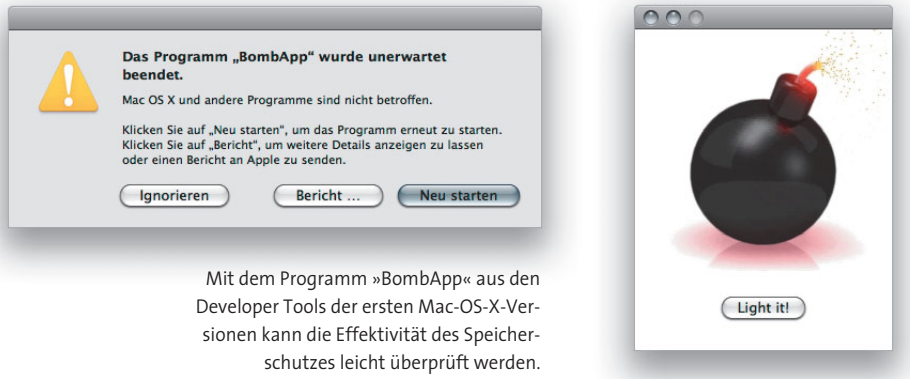
Speicher- und Prozessverwaltung

Der Arbeitsspeicher sowie sämtliche internen Chips und externen Geräte werden seitens der Computerhardware über Speicheradressen angesprochen. Die Zugriffe auf diese Speicheradressen jedoch müssen vom Betriebssystem verwaltet werden.

Das Betriebssystem ist aber auch für die Verwaltung der verschiedenen laufenden Prozesse zuständig.

Speicherverwaltung

Der Mach-Kern von Mac OS X besitzt eine der effektivsten Speicherverwaltungen in der gesamten Computerwelt. Er teilt jedem Prozess einen virtuellen Speicherbereich zu, der jeweils bis zu 4 GB groß sein kann – bei einem 64-bit-Prozess auf dem PowerPC G5 sogar bis zu 16 Exabytes bzw. beim Intel Core2 256 Terabytes (beim x86 ist wegen der prozessorinternen Speicherverwaltung derzeit nur ein virtueller 48bit-Adressraum, ein 65536stel, verfügbar). Der virtuelle Speicherbereich wird in kleine Teile (Pages) à 4 KB eingeteilt. Der Mach-Kernel versteht den Arbeitsspeicher und die Auslagerungsdateien auf der Festplatte als Cache für den virtuellen Speicher: Nur die tatsächlich benutzten Pages werden in den physikalischen Arbeitsspeicher des Mac geladen. Sollte die Anzahl der verwendeten Pages die Größe des physikalischen Arbeitsspeichers übersteigen, werden die am seltensten benötigten Pages auf die Festplatte ausgelagert – in Mac OS X in eine (oder mehrere) Auslagerungsdatei, die sich in einem unsichtbaren Ordner befindet (/var/vm/swapfile*X*). Unter allen geladenen Pages werden regelmäßig diejenigen aussortiert, die nicht mehr benötigt werden (Garbage-Collection). Wenn zwei identische Objekte im Arbeitsspeicher liegen (z.B. zwei Instanzen eines Programms oder wenn ein Objekt kopiert wird)



Mit dem Programm »BombApp« aus den Developer Tools der ersten Mac-OS-X-Versionen kann die Effektivität des Speicherschutzes leicht überprüft werden.

ist es sogar möglich, dass das nur eines der beiden Objekte komplett im Speicher liegt und das andere Objekt nur als Verweis.

Das Programm arbeitet immer nur mit seinem virtuellen Speicherbereich. Es erhält keine Informationen darüber, an welcher Stelle sich seine Objekte tatsächlich im Speicher befinden. Durch dieses System wird der Arbeitsspeicher immer effektiv genutzt. Programme, die besonders viel Arbeitsspeicher benötigen, stellen kein wirkliches Problem dar.

Speicherschutz

Da die virtuellen Speicherbereiche der verschiedenen Programme strikt voneinander getrennt sind, kann kein Programm versehentlich in den Bereich eines anderen schreiben. Ein abstürzendes Programm kann also kein anderes Programm oder gar das System mit in den Abgrund ziehen.

Multitasking

Der Mach-Kern des Mac OS X verwaltet außerdem auch die Prozessorzeit. Er teilt hochherrschaftlich jedem Prozess eine bestimmte Zeit zu. Ist diese Zeit abgelaufen, geht die Prozessorzeit an einen anderen Prozess über. Diese Form der Verwaltung der Prozessorzeit wird als präemptives Multitasking bezeichnet. (Im Gegensatz dazu steht das kooperative Multitasking – wie es beispielsweise im klassischen Mac OS verwendet wurde – bei dem der aktive Prozess von sich aus den Prozessor freigeben muss.)

Durch dieses Multitasking lässt sich beispielsweise in einem Programm weiter arbeiten, während ein anderes gestartet wird. Genauso gut kann ein abstürzendes Programm beendet werden, obwohl es nicht mehr reagiert.

Multiprocessing

Außerdem können vom Mach-Kernel Prozesse auf mehrere Prozessoren verteilt werden, ohne dass das Programm dafür speziell programmiert werden müsste.

Kommunikation zwischen Programmen

Das dem Mac OS X zugrunde liegende System aus Mach und BSD basiert auf Messaging. Es laufen viele unterschiedliche Prozesse, die alle mit dem Kernel, aber auch untereinander kommunizieren. Diese Eigenschaft kann von Anwender einfach mit den Funktionen der Shell und mit Shell-Scripts, aber auch aufwendiger

mit den UNIX-Skriptsprachen – PHP, TCL und Perl sind schon installiert – genutzt werden.

Eine zusätzliche Möglichkeit unter Mac OS X (und dem klassischen Mac OS) stellen die **»Apple Events«** dar. Möchte ein Mac-Programm eine Funktion eines anderen Programms in Anspruch nehmen, verschickt es als Anfrage ein AppleEvent. Sobald das zweite Programm diese Funktion ausgeführt hat, verschickt es ein AppleEvent als Antwort. So kann beispielsweise automatisch eine PPP-Verbindung geöffnet werden, wenn eine Adresse in einem Internetbrowser eingegeben wird. Auch »AppleScript« steuert Programmfunktionen über AppleEvents.

► 248ff
Praxis, AppleScript

► 57
Programm-Bundles

Das Terminal schafft mit dem AppleScript-Befehl »do script« eine Verbindung zwischen AppleEvents und dem UNIX-Messaging.

32-bit – 64-bit

◄ 31
Prozessoren, 32-bit – 64-bit

Nach der Einführung des G5-Prozessors vollzieht Apple einen schrittweisen Übergang von 32-bit nach 64-bit. In Panther (Mac OS X 10.3) konnte das System mit den 64-bit-Hardware-Adressen des PowerMac G5 umgehen. Seit Tiger können UNIX-Prozesse in der BSD-Umgebung im 64-bit-Modus arbeiten. Ein 64-bit-Programm mit Benutzeroberfläche musste aber aus zwei Teilen bestehen – einem 64-bit-Hintergrundprozess und einem 32-bit-Programm für die grafische Oberfläche. Unter Leopard stehen auch die Frameworks für die grafische Programmumgebung Cocoa in 64 bit zur Verfügung, sodass auch grafische Mac-OS-X-Programme in einer 64-bit-Version erstellt werden können.

Ein Programm in der falschen Architektur

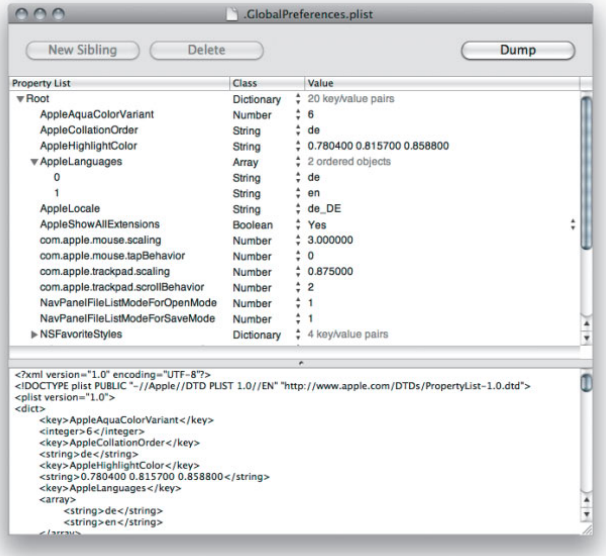
Trotzdem ist auch Leopard kein 64-bit-System "von Grund auf" (wie Steve Jobs behauptet). Der Kernel ist weiterhin 32 bit. Das ist insofern kein Problem, da der Kernel-Prozess recht wenig Speicher verbraucht und in absehbarer Zeit nicht mehr als 4 GB virtuellen Speicher zugeteilt bekommen muss. Beim PowerPC ist das auch vom Design her so vorgesehen – der 32-bit-Kernel kann im 64-bit-Modus des Prozessors ausgeführt werden. Beim x86 jedoch ist der 32-bit-Kompatibilitätsmodus ausdrücklich nur für Benutzerprozesse gedacht, nicht für Kernel-Prozesse. Apple setzt sich hier über Vorgaben Intels (bzw. AMDs) hinweg.

Universal Binarys

Mac OS X bietet die Möglichkeit, Programme für PowerPC und für x86 zu einem einzigen Programm zusammenzufassen. Bei einem Großteil der Dateien im Programm-Bundle handelt es sich um Dateien für die grafische Benutzeroberfläche des Programms, wie Fenster, Menüs, Symbole etc. Daten dieser Art sind von sich aus plattformunabhängig. Nur die Binärdatei in Contents/MacOS enthält plattformabhängigen Code.

Mit den Universal-Binarys kann aber auch diese Datei plattformunabhängig gestaltet werden. In der Datei ist dann sowohl der Code für den Intel-Prozessor als auch der für den PowerPC-Prozessor enthalten, bei 64-bit-fähigen Programmen jeweils in 32- und 64-bit-Version. Beim Kompilieren des Programms wird der Code hintereinander in die Datei geschrieben. Im Datei-Header finden sich Informationen, für welche Architekturen Code vorhanden ist und wo in der Datei sich der jeweilige Code findet.

In der privaten allgemeinen Voreinstellungsdatei ».globalpreferences.plist« werden u.a. die Spracheinstellungen gespeichert. Der »Property List Editor« aus den Developer Tools stellt den Inhalt übersichtlich dar. Im unteren Teil sehen Sie den originalen XML-Text.



Voreinstellungen

Voreinstellungen von Programmen werden in Mac OS X für jeden Benutzer separat in einzelnen Dateien im Ordner »*Privat*/Library/Preferences« angelegt. (Die Voreinstellungen der Programme, die vor der Anmeldung der Benutzer gestartet werden, befinden sich im Ordner »/Library/Preferences«.) Beim Starten des Programms werden die Voreinstellungsdateien gelesen und die entsprechenden Voreinstellungen vorgenommen. Jedes Programm kann eine oder mehrere Voreinstellungsdateien anlegen. Diese können Sie in der Regel am Namen ihrem Erzeuger zuordnen.

Voreinstellungsdateien können dabei – wie jedes andere Objekt – aus dem Ordner herausbewegt oder in den Papierkorb gelegt und gelöscht werden. Das entsprechende Programm kann dann die Voreinstellungen nicht lesen und legt eine neue Voreinstellungsdatei mit den im Programm gespeicherten Standard-Voreinstellungen an.

Voreinstellungen von Mac-OS-X-Programmen werden als Textdateien in XML verfasst. Sie erhalten den Namen des Programmpakets (z.B. »com.apple calculator.plist« für den »Rechner«). Sie können z.B. im »Property List Editor« aus den Developer Tools bearbeitet werden. Mit dem Befehl »plutil -lint« können sie in der Shell auf Syntaxfehler hin überprüft werden. Ab Mac OS X Tiger werden die Dateien beim Zugriff durch das System komprimiert.

284 ▸
Praxis, Voreinstellungsdatei defekt

Voreinstellungen können auch durch Kopieren der Voreinstellungsdatei in den Preferences-Ordner eines anderen Benutzers oder gar eines anderen Mac übertragen werden. (Carbon-Programme, die auch unter dem klassischen Mac OS laufen, verwenden dasselbe Format wie im klassischen Mac OS. So können Sie auch Voreinstellungen von Programmen, die sie schon unter dem klassischen Mac OS verwendet haben, in Mac OS X übernehmen.)

Voreinstellungen von UNIX-Programmen werden in Config-Dateien, reinen ASCII-Textdateien, gespeichert.

256 ▸
Praxis, Konfigdateien

▸ 264
Praxis, Speichermedien formatieren

Massenspeicherverwaltung

Bei Festplatten, CDs und DVD, USB-Sticks etc. spricht man von Massenspeichern. Diese werden zum einen vom Benutzer verwaltet, der Dateien anlegt und diese in Ordner legt. Zum anderen aber muss auch das Betriebssystem die Massenspeicher verwalten, um auf die Daten zugreifen zu können, wenn sie auf den Bildschirm gebracht oder vom Prozessor verarbeitet werden sollen.

▸ 263
Praxis, Dateisysteme
▸ 265
Praxis, Partitionsschema

Partitionsschemata

Am Anfang eines jeden Datenmediums findet sich ein Eintrag über die auf dem Medium vorhandenen Volumes – die sogenannte Partitionstabelle. Dabei verwenden die unterschiedlichen Rechnerarchitekturen unterschiedliche Formate. Der Mac mit PowerPC verwendet die Apple-Partitionstabelle (Apple Partition Map, APM), der normale PC mit BIOS die MBR-Partitionstabelle (Master Boot Record) und der Mac mit Intel-Architektur die GUID-Partitionstabelle (GPT), die in Intels EFI-Standard vorgegeben ist. MBR und GPT können zusammen auf einem Medium existieren. Alle Macs mit Mac OS X seit 10.4.4 können mit dem Apple-, dem GUID- und dem MBR-Partitionsschema umgehen. Macs mit älteren Systemen aber evtl. nur mit dem Apple- und dem MBR-Partitionsschema. Das Apple-Partitionsschema ist inkompatibel mit DOS-formatierten Volumes. DOS-formatierte Volumes sind nur auf Festplatten mit dem MBR- oder dem GUID-Partitionsschema möglich. Damit jedoch ein Volume auf dem Medium in dem jeweiligen Mac startfähig sein kann – zu dem Zeitpunkt, an dem beim Systemstart das erste Mal auf die Festplatte zugegriffen werden muss, läuft Mac OS X noch nicht – muss für den PowerPC-Mac das Apple-Partitionsschema und für den Mac mit Intel-Architektur das GUID-Partitionsschema verwendet werden. Entgegen Apples offizieller Aussage startet ein Mac mit Intel-Architektur aber auch von mit dem Apple-Partitionsschema partitionierten Medien, wie der Mac OS X DVD – oder einer Firewire-Festplatte, auf die mit einem PowerPC-Mac Mac OS X installiert wurde.

Das Dateisystem

Zum Verwalten der Daten auf der Festplatte und anderer Speichermedien benötigt jedes Betriebssystem ein Dateisystem. Dieses Dateisystem ist nicht zu verwechseln mit dem Dateisystem, welches der Benutzer z.B. über den Finder zu sehen bekommt. Es bietet jedoch mit seiner Struktur die Grundlage für die Hierarchien des Finders.

Wie jedes Betriebssystem benutzt Mac OS X ein eigenes, an seine Bedürfnisse und Fähigkeiten angepasstes Dateisystem. Das Dateisystem von Mac OS X trägt den Namen »**HFS+**« (Hierarchical File System). HFS+ organisiert die Daten auf der Festplatte über so genannte B*-Bäume (balancierter Baum), die besonders schnelles Auffinden der Einträge ermöglichen. Jedes Objekt, egal ob Ordner oder Datei, besitzt eine ID-Nummer (Catalog Node ID, CNID). Wie bei einer Seriennummer erhält jedes neue Objekt eine neue ID. IDs von gelöschten Objekten werden nicht mehr vergeben (es sei denn, die größte mögliche Zahl für die ID-Nummer wurde überschritten), eine neue Nummerierung wird erst begonnen, wenn das Volume initialisiert wird. Zugriffe auf Dateien und Ordner erfolgen anhand der ID-Nummer – nicht anhand des Dateinamens. Im Knoten (die Einträge im B-Baum) ist zu jedem Objekt als zusätzliche

Eine Festplatte ist durch magnetische Linien in kleine Blöcke aufgeteilt.

Information die ID des Ordners gespeichert, in dem es sich befindet. So kann der Finder die gesamte Hierarchie zurückverfolgen, in der sich ein Objekt befindet. Für den Zugriff auf die in den Dateien gespeicherten Daten, ist im Blattknoten die Position der Daten auf dem Datenmedium gespeichert.

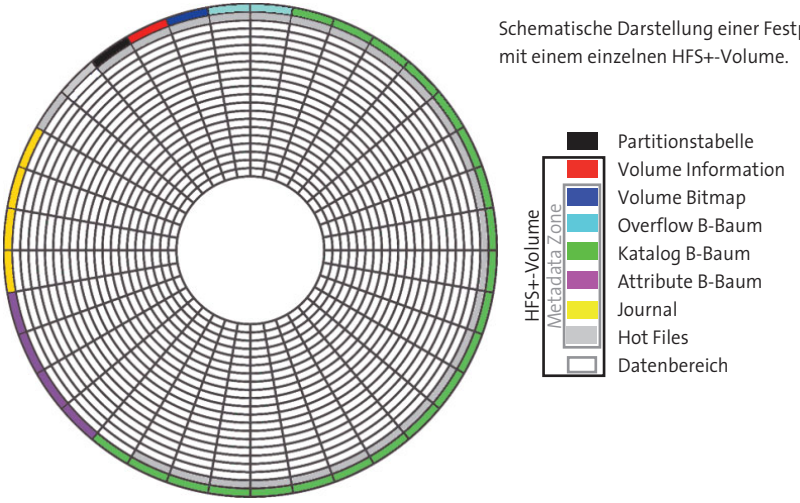
Auch **UNIX-Dateisysteme** verwenden Dateinummern (hier iNode-Nummern) für die Verwaltung der Dateien, benutzen dabei jedoch einfache, über die Festplatte verteilte Tabellen. Nur das **MS-DOS-Dateisystem FAT** verwaltet die Dateien anhand ihrer Namen und Pfade, in einer einzigen großen Tabelle.

Blöcke und Zuteilungsblöcke

Festplatten werden mittels magnetischer Linien in Blöcke eingeteilt, die jeweils 512 Bytes Daten enthalten können. Da aber zu viele Blöcke die Arbeit des Dateisystems verlangsamen und auch einer Fragmentierung vorgebeugt werden muss, fasst das Dateisystem als Kompromiss zwischen Platz und Geschwindigkeit mehrere Blöcke auf der Festplatte zu Zuteilungsblöcken zusammen (auch Cluster genannt). Bei Festplatten, welche in HFS+ formatiert sind, z.B. ist die Blockgröße 4 kB, bei einer unter Mac OS X gebrannten Daten-CD 2 kB. Im ungünstigsten Falle belegt damit eine 1 Byte große Datei 4 kB auf der Festplatte. HFS+ benutzt für die Adressierung der Zuteilungsblöcke eine 32 bit lange Zahl. Es kann also bis zu 4.294.967.296 Blöcke adressieren. Beim MS-DOS-Dateisystem FAT32 sind 4 KB die kleinste Clustergröße, für Volumes kleiner, als 8 GB. Bei größeren Volumes sind die Cluster 8, 16 oder bis zu 32 kB groß.

HFS+

Das HFS+-Dateisystem besteht aus mehreren B*-Baum-Dateien. Im größten, dem Katalog (Catalog File), wird der Aufenthaltsort jeder einzelnen Datei auf der Festplatte eingetragen. In einer weiteren Datei (Extents Overflow File, in einer älteren Version des Festplatten-Dienstprogramms auf deutsch »Zusatzdatei für Dateiaufbau« genannt) wird für fragmentierte Dateien – Dateien, die nicht als ein zusammenhängendes Stück auf der Festplatte liegen – verzeichnet, an welcher Stelle sich weitere Teile dieser Datei befinden. Hinzu kommt noch eine B-Baum-Datei, in der erweiterte Attribute, verzeichnet werden können.



Schematische Darstellung einer Festplatte mit einem einzelnen HFS+-Volume.

Am Anfang des Volumes steht der Header – die Volume-Informationen. In diesem Bereich wird das Volume als Mac-Volume identifiziert, sind der Name des Volumes und verschiedene andere Informationen abgelegt, z.B. die Größe und Position der Dateisystems-Dateien, die Nummer, die das nächste Objekt bekommt, und die Objektnummer des aktiven Systemordners (des »blessed Folders«).

HFSX

Mit Panther neu hinzugekommen ist HFSX, das gegenüber HFS+ weitere Möglichkeiten bietet. Implementiert ist zurzeit aber nur die Möglichkeit, das Dateisystem Case-sensitive anzulegen, d.h. ein gleicher kleiner und großer Buchstabe werden unterschiedlich behandelt.

▶ 55 Case-sensitivity

Journal

Seit Mac OS X 10.2.2 kann für HFS+-Volumes auch ein Journal aktiviert werden. Wenn das Journal aktiviert ist, werden die letzten Änderungen am Dateisystem noch einmal zusätzlich in eine Datei (das Journal) geschrieben. Bei einem eventuellen Absturz kann der zuletzt geänderte Teil des Dateisystems anhand der Daten im Journal überprüft und repariert werden. Ein Durchlauf von fsck, der das gesamte Dateisystem überprüft, ist so nicht nötig.

▶ 274 Praxis, Journal Aktivieren

Adaptive Hot Files Clustering

Mit Panther wurde für HFS+-Volumes ein System eingeführt, das die Zugriffszeiten auf Dateien auf der Festplatte optimiert. Auf Festplatten befinden sich tausende Dateien, kleine und große. Aber nur auf wenige, meist kleine Dateien wird ganz besonders häufig zugegriffen. Außerdem wird sehr häufig auf das Dateisystem zugegriffen.

Wenn also die Dateien, auf die besonders häufig zugegriffen wird, nahe am Dateisystem liegen, erspart das dem Schreib-Lesekopf viele Wege. Damit werden die Zugriffe beschleunigt. Mac OS X führt daher in einer eigenen Datenbank (/.hotfiles.btree) Buch über die Zugriffe und verschiebt Dateien, auf die ganz besonders häufig zugegriffen wird (die »heißesten« Dateien), in einen Bereich direkt hinter dem Dateisystem. Umgekehrt werden aber auch Dateien, auf die nicht mehr so häufig zugegriffen wird,

wieder in den normalen Bereich der Festplatte verschoben. Diese Funktion nennt sich »adaptive hot files clustering«.

Metadata Zone
Beim Initialisieren eines Volumes reserviert Mac OS X am Anfang des Volumes den Platz, der für das Dateisystem, das Journal und die Hot-Files benötigt wird. Dieser Bereich wird »Metadata Zone« genannt.

Datei-Fragmentierung
HFS+ verhindert die Fragmentierung von Dateien schon von vorne herein. Das Volume Bitmap – ein Abbild des Volumes, in dem belegte und unbelegte Blöcke anhand der »Farbe« (schwarz bzw. weiß) erkannt werden können – ermöglicht es dem Dateisystem, einen Bereich auf der Festplatte zu finden, der groß genug für eine zu schreibende Datei ist. Außerdem verwendet HFS+ sogenannte Extents. Für eine spätere Erweiterung wird beim Schreiben einer Datei ein größerer Bereich reserviert, als eigentlich nötig. Sollten diese Maßnahmen nicht reichen, greift die On-the-fly-Defragmentierung. Beim Zugriff auf Dateien bis zu einer bestimmten Größe überprüft Mac OS X, zu welchem Grad diese fragmentiert ist. Wird dabei ein bestimmter Wert übertroffen, wird die Datei automatisch defragmentiert, indem sie komplett an einen freien Platz auf der Festplatte verschoben wird.

Andere Dateisysteme
Neben HFS+ kann Mac OS X mit einigen weiteren Dateisystemen wie dem älteren Mac-Dateisystem HFS, dem UNIX-Dateisystem UFS, den Windows-Dateisystemen FAT und NTFS (NTFS nur lesend), sowie UDF (Universal Disk Format) für DVDs, ISO 9660/Joliet auf Windows-CD-ROMs usw. umgehen.

Sie alle besitzen unterschiedliche Eigenschaften besitzen. Damit weder die Programme noch die Anwender die Besonderheiten der unterschiedlichen Dateisysteme berücksichtigen müssen, legt Mac OS X über alle verwendeten Dateisysteme ein virtuelles Dateisystem und konvertiert die Dateinamen und Datenstrukturen für das tatsächlich verwendete Dateisystem im Hintergrund. In das virtuelle Dateisystem werden auch die Netzwerk-Dateisysteme eingebunden, die von den Netzwerkprotokollen der File-Server erzeugt werden.

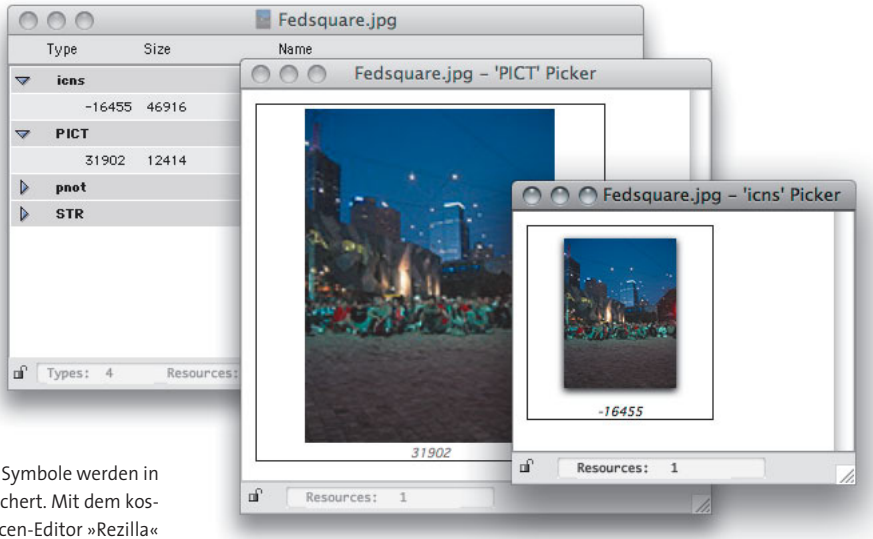
Bei Audio-CDs generiert Mac OS X sogar ein Dateisystem, obwohl auf der Audio-CD gar keines vorhanden ist (eine Audio-CD enthält nur einen durchgehenden Datenstrom). Dafür benutzt es die Track-Informationen im TOC (Table of Contents).

275 ▸
Praxis, Defragmentierung

Case-sensitivity
HFS+ ist »Case insensitive but Case respecting«. Das Dateisystem speichert zwar die großen und kleinen Buchstaben, verwendet beim Suchen aber einen Algorithmus, der sie behandelt, als wären sie gleich.

UNIX-Dateisysteme jedoch sind »Case sensitive«, sie unterscheiden zwischen großen und kleinen Buchstaben. Bei »Text.txt« und »text.txt« handelt es sich dort also um zwei verschiedene Dateien. Mit HFSX gibt es die Möglichkeit ein Case sensitives Mac-Dateisystem anzulegen.

Daten und Ressourcen, benannte Zweige
Eine klassische Mac-Datei ist in der Regel in zwei »Zweige« (Forks) unterteilt: in den Datenzweig und den Ressourcenzweig. Seit Leopard verwendet Mac OS X auch die »Named Forks« (benannte Zweige), welche bereits beim Entwurf von HFS+ vorgesehen waren.



Vorschaubilder und Symbole werden in Ressourcen gespeichert. Mit dem kostenlosen Ressourcen-Editor »Rezilla« können Ressourcen geöffnet werden.

▸ 58 **Objekt-attribute**

Der Datenzweig enthält reine Textinformationen oder Binärdaten, wie die Dateien von anderen Betriebssystemen auch.

Der Ressourcenzweig ist eine Besonderheit des Mac OS und des Mac-Dateisystems HFS. Ressourcen enthalten beispielsweise Dateiinformationen, Symbole oder Vorschaubilder. Die Ressourcen werden in einer Art Datenbank verwaltet, sie entsprechen einer vorgegebenen Struktur. Sie können daher auch von anderen Programmen gelesen werden.

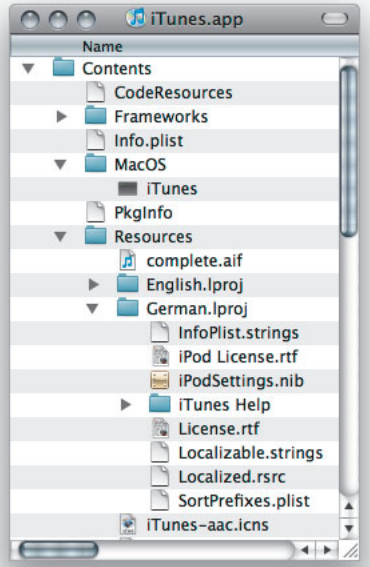
Bei klassischen Mac-Programmen kamen noch Menüeinträge, Dialogfenster und deren Inhalt usw. hinzu. Auch einige Programme, die unter Mac OS und unter Mac OS X laufen (Carbon CFM) machen weiterhin Gebrauch vom Ressourcenzweig. Reine Mac-OS-X-Programme jedoch verwenden den Ressourcenzweig nicht.

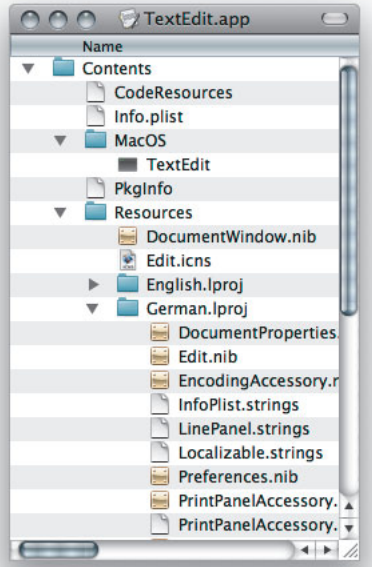
In Named Forks werden erweiterte Dateieigenschaften gespeichert. Sie sind nach dem umgekehrten DNS-Schema benannt. (z.B. com.apple.TextEncoding für die Text-Kodierung in einer Textdatei, siehe Bild Seite 60). Auch die Resourcen und klassischen Mac-Attribute werden in Leopard in benannte Zweige umgesetzt.

Bundles

Ein Bundle ist eigentlich nur ein Ordner, der jedoch im Finder als Datei erscheint. Ein Ordner wird dadurch zu einem Bundle, dass seinem Namen ein Suffix (z.B. ».app« bei einem Programm) hinzugefügt wird. Innerhalb des Ordners werden die Daten nach einer bestimmten Struktur abgelegt. Diese Struktur wird in anderen Betriebssystemen (Windows, Linux, Mac OS 8) sichtbar, unter Mac OS X sieht der Anwender jedoch nur ein Programm, das er per Doppelklick starten kann. Mit dem Kontextmenü-Befehl »Paketinhalt anzeigen« können Sie sich den Paketinhalt als Ordner anzeigen lassen.

Neben den Programm-Bundles gibt es in Mac OS X noch weitere Typen von Bundles: Kernel-Extensions und Frameworks, sowie verschiedenste andere Bundle-Typen. So packt beispielsweise TextEdit RTF-Dateien mit Bildern





Die Programm-Bundles des Cocoa-Programms TextEdit und des Carbon-Programms iTunes.

in ein .rtfd-Bundle, und auch die Dateien der Apple-Textverarbeitung »Pages« sind Bundles. Solche Bundles werden erst als Bundle erkannt, wenn das entsprechende Programm auf dem Mac vorhanden ist. Sonst werden sie als normale Ordner angezeigt.

Programmbundles

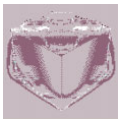
Bei Mac-OS-X-Programmen werden die Daten und Ressourcen des Programms in einzelnen Dateien in den Unterordner »Contents« des .app-Bundles gepackt. Im Ordner »Mac OS« befindet sich der Datenteil, im Ordner »Resources« sind in einzelnen Dateien die verschiedenen Ressourcen. Das Programm-Symbol und die Datei-Symbole beispielsweise werden in .icns-Dateien gespeichert. Sprachspezifische Ressourcen der unterschiedlichen Sprachversionen werden in den .lproj-Ordner der jeweiligen Sprache gepackt. Hinzu kommen noch Dateien mit Informationen zum Inhalt des Bundles und zu den Dokumententypen (Info.plist), der Type- und Creator-Kennung (PkgInfo) und der Versionsinformation (Version.plist). Alle Dateien des Bundles befinden sich im Datenzweig.

Carbon- und Cocoa-Programme unterscheiden sich kaum. Carbon-Programme verwenden zum Teil die in .rsrc-Dateien gespeicherte Ressourcen-Struktur des Mac OS. Die vom Interface-Builder erzeugten Ressourcen (Menüs, Fenster etc.) der Cocoa-Programme aber auch neuer Carbon-Programme sind in .nib-Dateien gespeichert. Bilder u. Ä. werden als einzelne Dateien gespeichert.

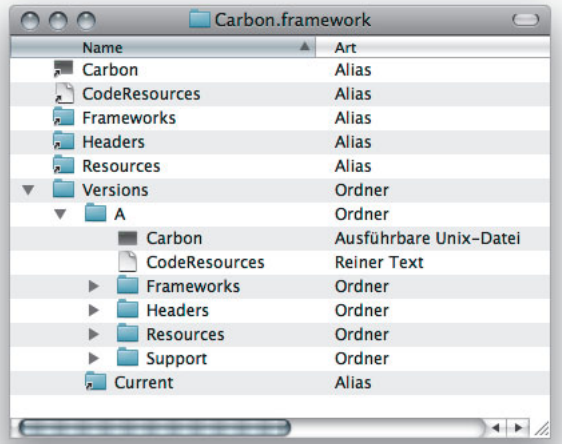
Das allgemeine Programmsymbol

Erweiterungen

In Mac OS X werden Treiber für Komponenten des Mainboards und für PCI-Karten oder bestimmte Protokolle in Form von Erweiterungen (Kernel-Extensions, KEXT) dem Systemkern hinzugefügt. Die Erweiterungen befinden sich im Ordner »/System/Library/Extensions«. Eine Erweiterung besteht aus mehreren Dateien in einem .kext-Bundle – dem Treiber, der Infodatei und den Ressourcen.



Das Symbol einer Kernel-Extension



Frameworks können in verschiedenen Versionen zusammengefasst werden. Die Aliase auf der obersten Ebene verweisen auf die entsprechenden Ordner der aktuellen Version, der Alias »Current« verweist auf den Ordner »A«, denn dies ist die aktuelle Version.

Frameworks

Routinen, die von mehreren Programmen benutzt werden können, werden nicht in den Programmcode integriert, sondern in Form von Bibliotheken als eigene Datei gespeichert und dynamisch mit dem Programmcode verlinkt. (Im klassischen Mac OS heißen sie Libraries oder Shared Libraries, unter Windows sind es die DLLs.)

▸ 154f Praxis, Etiketten

Im Mac OS X werden Bibliotheken als Frameworks organisiert. Wie in anderen Bundles befinden sich auch hier die Daten und die Ressourcen in getrennten Dateien in einem gemeinsamen Ordner (.framework). In einem Framework-Bundle können sich mehrere Versionen einer Bibliothek befinden, sodass Probleme mit der Kompatibilität verschiedener Programme und unterschiedlicher Versionen (wie sie z.B. unter Windows nicht unüblich sind) ausgeschlossen werden. Die aktuelle Version wird dabei besonders markiert. Frameworks werden im Ordner »/System/Library/Frameworks« gespeichert.

▸ 60 Type/Creator

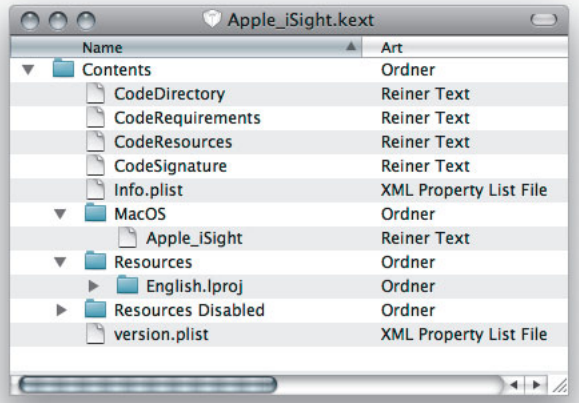
Objektattribute

Der Mac gibt jedem Objekt einen Infostring mit einer Reihe von Informationen mit. Die bekanntesten sind natürlich der Objektname, das Erstellungsdatum, das Änderungsdatum und die Größe. Bei UNIX-Systemen kommen dazu die Dateirechte. Im Mac-Dateisystem HFS+ enthalten Objekte zusätzlich zu diesen Informationen noch die »Finder Flags«, einige Bits, die weitere Informationen wie z.B. Sichtbarkeit oder Etikett kodieren. Auch die Benutzerrechte sind als Objektattribute im Dateisystem angelegt. Alle diese Informationen werden im Mac-Dateisystem HFS+ in den Knoten des Dateisystems gespeichert.

Auch der Dateityp kann beim Mac als Information im Dateisystem gespeichert werden. Er wird mit der Type- und der Creator-Kennung kodiert.

Objektnamen

Jedes Objekt im Mac OS X trägt einen Namen. Die dafür verwendeten Objektnamen dürfen bis zu 255 Zeichen lang sein. Jedes Zeichen darf verwendet werden – mit einer Ausnahme: dem



Der Aufbau einer Kernel-Extension

Doppelpunkt. Er wird zur Trennung der Pfadnamen benutzt.

Der Finder erzeugt trotzdem bei dem Versuch, einen Doppelpunkt einzugeben, keine Fehlermeldung – er ersetzt ihn einfach durch einen Strich. Der Doppelpunkt wird sogar dann automatisch durch einen Strich ersetzt, wenn ein Name, der einen Doppelpunkt enthält, durch »Kopieren und Einsetzen« vergeben wird. Auch in den Sichern-Dialogen lässt sich der Doppelpunkt nicht eintippen.

Dateien dürfen beliebig benannt werden. Dateityp-Suffixe, wie Windows sie benutzt, braucht der Mac nicht unbedingt. Diese Funktionen können in Mac OS X die Type- und Creator-Kennungen übernehmen. Trotzdem sollten Sie Suffixe anhängen, damit beispielsweise bei der Übertragung über das Internet noch zu erkennen ist, welcher Art die Datei ist.

Ordnerlokalisierungen

Im Finder und in den Öffnen- und Sichern-Dialogen der Programme werden die vom System angelegten Ordner lokalisiert angezeigt. Das geschieht mithilfe der Vorgaben in der Datei »SystemFolderLocalizations.strings« für die ausgewählte Sprache (auf deutsch in »/System/Library/CoreServices/SystemFolderLocalizations/de.lproj«). Mit einer unsichtbaren Hinweis-Datei ».localized«, die in dem zu lokalisierenden Ordner liegt, wird das System angewiesen, den Namen dieses Ordners lokalisiert anzuzeigen. Programme werden anhand des Namens in der Datei »Infoplist.strings« des Sprachpakets lokalisiert.

Volumenamen

Beim Mac tragen neben Dateien auch alle Volumes einen Namen. Logische Laufwerke wie »C:« bei Windows gibt es nicht. Volumes, die keinen Namen haben, erhalten den Namen »Ohne Titel«. Der Name eines HFS-Volumes darf höchstens 63 Zeichen lang sein. Theoretisch können alle Volumes denselben Namen haben, was aber aus Gründen der Übersichtlichkeit nicht empfehlenswert ist.

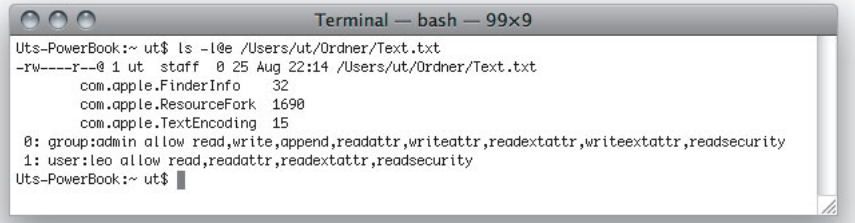
212ff ▸ Praxis, Datenaustausch mit dem Windows-PC

Dateirechte

Auch die Dateirechte sind Objektattribute des Dateisystems. Die UNIX-Dateirechte werden im Katalog gespeichert. Mit diesen können dem Eigentümer, einer Gruppe und jedermann Rechte für Dateien und Ordner zugeteilt werden. Daneben ist eine verfeinerte Zugriffsteuerung

129 ▸ Praxis, Dateirechte bestimmen

In der Shell werden die »Named Forks«und die »Access Controll Lists« von »ls« mit den Optionen »@« bzw. »e« angezeigt. (Hier sehen Sie die Ausgabe des Befehls »ls -l@e«.)



über ACLs (Access Control Lists – Zugangskontrolllisten) möglich. Hier können in Form einer Liste Rechte für weitere Benutzer und Gruppen definiert werden.

Die Dateisysteme FAT und HFS kennen keine Dateirechte.

Type und Creator

Auf einem Computer werden verschiedenste Dateitypen verwendet. Zwar bestehen alle Dateien aus Folgen von Einsen und Nullen; Systemdateien, Programme oder Dokumente verhalten sich jedoch ganz unterschiedlich.

Es wird also eine Methode benötigt, mit der sich die einzelnen Dateitypen auf Anhieb unterscheiden lassen. Der einfachste Weg ist, einen Teil des Dateinamens zu verwenden. Diese Methode wird beispielsweise unter Windows in Form eines drei Zeichen langen Suffixes verwendet (Dateiname.xxx). Windows-Dateien brauchen unbedingt dieses Suffix, damit das System sie einem Programm zuordnen kann.

Diese Lösung ist unflexibel, da sie voraussetzt, dass immer nur ein Programm einen Dateityp erzeugen kann. Außerdem kann der Anwender den Dateityp versehentlich verändern und so die Datei unbrauchbar machen.

Der Mac geht hier einen anderen Weg. Im Dateisystem sind im Infostring der Datei zwei voneinander unabhängige Kennungen verzeichnet. (Im Mac-OS-X-Programm-Bundle befinden sich die Kennungen in der Datei »PkgInfo«) Die Type-Kennung kodiert den Dateityp, die Creator-Kennung das Programm, das diese Datei erzeugt hat. Dadurch können Dokumente gleichen Typs eindeutig einem Erzeugerprogramm zugeordnet werden.

Ein Photoshop-EPS und ein FreeHand-EPS können beispielsweise beide gleichzeitig per Doppelklick mit dem jeweiligen Programm geöffnet werden.

Die Type- und die Creator-Kennung bestehen jeweils aus vier Buchstaben. Alle Zeichen können verwendet werdenk, große und kleine Buchstaben werden unterschiedlich behandelt. Hier einige Beispiele für Type- und Creator-Kennungen:

- **Type-Kennungen:** Eine Textdatei bekommt die Kennung »TEXT«, ein Pict-Bild »PICT« oder ein Programm »APPL« (für Application Program).
- **Creator-Kennungen:** »ttxt« steht für das Erzeugerprogramm »TextEdit« (»ttxt« steht eigentlich für »TeachText«, später bekannt als »SimpleText«), »GKON« für den »GraphicConverter« oder »8BIM« für »Photoshop«.

Anhand dieser beiden Angaben kann der Finder Beziehungen zwischen Programmen und Dokumenten herstellen und in der Listendarstellung oder in der Infobox eine Angabe über die Objektart machen. So ist eine Datei mit Type »PICT« und Creator »ttxt« ein SimpleText-Bild, aber mit Creator »GKON« ein GraphicConverter-PICT-Bild.

Alle Programme bekommen die Type-Kennung »APPL« und ihre Creator-Kennung. In Mac OS X wird diese Kennung jedoch nicht mehr konsequent eingesetzt, einige Programme vergeben keine Type-Creator-Kennungen mehr. Außerdem besitzen Dateien, die von anderen Plattformen kommen, diese Codes nicht. Mac OS X besitzt deshalb die Fähigkeit, Dateitypen anhand ihres Suffixes oder ihres Types zuzuordnen, aber auch anhand einer Kombination von Type bzw. Suffix mit dem Creator.

LaunchServices

Mac OS X führt über die Verbindungen zwischen Dokumenten und Programmen Buch. Dafür baut es über die »LaunchServices« eine Datenbank auf, in der sich Informationen darüber befinden, welches Programm welchen Dateityp öffnen kann und welche Symbole die Programme und ihre Dokumente bekommen.

Jedes Mac-OS-X-Programm enthält in seinem Bundle eine Datei – »Info.plist« (Informations Property List) – in der verzeichnet ist, welche Art von Dokumenten es öffnen kann. Bewegt der Anwender jetzt ein Dokument mit einer geeigneten Type-Kennung oder Dateinamenserweiterung auf ein Programmsymbol, aktiviert der Finder das Programmsymbol. Sobald der Anwender nun das Dokumentensymbol über dem Programmsymbol loslässt, öffnet das Programm das Dokument. Ist der Typ des Dokuments nicht verzeichnet, wird das Programmsymbol auch nicht aktiviert.

In der Datei »Info.plist« kann der Finder außerdem erkennen, welche Symbole den Dateien zugewiesen werden sollen, die von diesem Programm erzeugt wurden, und wie dieser Dateityp benannt werden soll. Der Mac-OS-X-Finder speichert diese Informationen für jeden Benutzer einzeln. Dafür liest er zuerst nur die Standard-Ordner für Programme aus. Weitere Informationen werden in den verschiedenen Ordnern erst ausgelesen, wenn der Anwender in einen neuen Ordner navigiert.

Uniform Type Identifiers

Seit Mac OS X 10.4 werden von den Launchservices verschiedene Suffixe und Types – aber auch die MIME-Typen, die im Internet verwendet werden – die gleiche Dateitypen kodieren, nach einem ausgeklügelten System gegen einen »Uniform Type Identifier« (UTI) aufgelöst (z.B. die Suffixe ».txt« und ».text« zu »public.plain-text« oder das Suffix ».pdf« und der Type »PDF « zum »com.adobe.pdf«).

Ressourcen und Objektattribute in flachen Dateisystemen

Das HFS+-Dateisystem bietet dem Mac mit dem Ressourcenzweig und den Objektattributen bzw. den benannten Zweigen Besonderheiten, die andere Dateisysteme nicht kennen. Daher muss Mac OS X zu einem Trick greifen, um die Daten, die dort gespeichert sind auch auf anderen Dateisystemen zu erhalten: Das AppleDouble-Format.

Wird die Datei auf ein flaches Dateisystem ohne Zweige kopiert, wird unter Mac OS X zusätzlich zur Datei, die den Datenzweig enthält, eine zweite Datei mit gleichem Namen, aber mit dem Präfix »._« angelegt. In diese ._ -Datei werden die Ressourcen und Attribute, die im Dateisystem HFS+ in den Zweigen gespeichert sind, transferiert.

Ressourcen und Objektattribute und andere Betriebssysteme

Wird das Kopieren oder die Übertragung einer Mac-Datei mit verschiedenen Zweigen nicht von Mac OS X aus initiiert, ergibt sich ein Problem: Andere Betriebssysteme (Windows, UNIX) und normale Übertragungsprotokolle

212ff ▸
Praxis, Datenaustausch mit dem Windows-PC

Spotlight sammelt die Metainformationen aus Dateien. In diesem Infofenster werden die EXIF-Informationen einer Bilddatei aufgelistet.

erkennen die Verbindung zwischen dem Datenzweig und den anderen Zweigen einer Mac-Datei nicht. In der Regel wird nur der Datenteil kopiert, der Ressourcenteil und die Attribute aber gehen ganz verloren.

Wenn Dateien über das Internet verschickt werden sollen – das ja nur zu einem kleinen Teil aus Macs besteht und zum größten Teil aus »normalen« UNIX-Rechnern –, muss eine Verbindung zwischen den Zweigen geschaffen werden. Dafür werden Mac-Dateien vor dem Verschicken kodiert. Sie werden so zu reinen Text- bzw. Binärdaten, die später wieder zu Mac-Dateien dekodiert werden können.

▶ 203 Praxis, FileSharing, Daten kodieren

Spotlight

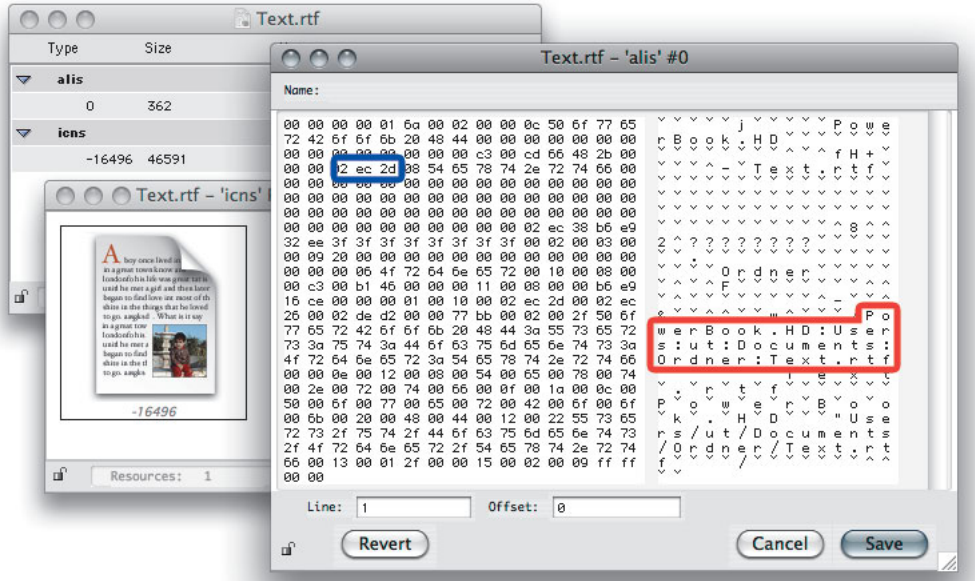
Neben den Objektattributen im HFS+-Dateisystem gibt es innerhalb vieler Dateien weitere Metadaten. Diese stellen Informationen über eigentlichen Daten in der Datei bereit – z.B. die EXIF-Daten in Bilddateien, ID3-Tags in MP3-Musikdateien aber auch Copyright-Informationen. Diese Daten werden von Spotlight zusammen mit den Objektattributen in einer Datenbank gesammelt. In einer weiteren Datei wird ein Index der Textinhalte von Textdateien angelegt. Diese Daten stehen damit allen Spotlight-fähigen Programmen zur Verfügung. Selbst in der Kommandozeile kann über die Befehle »mdls« und »mdfind« auf Spotlight zugegriffen werden. Mit den Befehlen »mdimport« und »mdutil« kann Spotlight zum erneuten Indizieren veranlasst werden – komplett oder nur mit einem bestimmten Importfilter – sowie ein- und ausgeschaltet werden etc.

▶ 140ff Praxis, Spotlight, Dateien finden

Alias, Symbolic Links und Hardlinks

Seit System 7 bietet das Mac OS dem Anwender die Möglichkeit, Alias-Dateien anzulegen. Diese Dateien sind Verweise auf eine Originaldatei oder einen Ordner.

Aliase können von jedem beliebigen Objekt angelegt werden, auch von Objekten, die sich auf einem anderen Volume oder irgendwo im Netzwerk befinden. Aliase besitzen dieselben Eigenschaften wie ihr Original. Deshalb können



In der Ressourcen eines Alias ist der Pfad zum Original (rot) und die ID des Ordners, in dem es sich befindet (blau) gespeichert, sowie das passende Symbol.

sie beispielsweise genauso wie das Original zum Öffnen per Drag&Drop benutzt werden. Aliase sind zu erkennen an einem kleinen Pfeil an der unteren linken Ecke des Symbols.

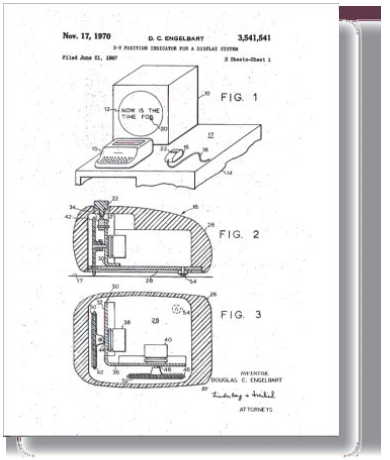
Die Verbindung zum Original wird in der »alis«-Ressource der Alias-Datei als Pfad abgespeichert. Im Infofenster wird dieser Pfad angezeigt. Ergänzend wird hier aber auch die ID-Nummer des Ordners, in dem sich die Originaldatei befindet, gespeichert. So wird das Original auch dann wiedergefunden, wenn es verschoben und umbenannt wurde. Das funktioniert sogar, wenn sich das Original auf einem anderen Volume befindet, das während des Verschiebens und Umbenennens nicht gemountet war. (Einzige Ausnahme: Wenn ein Objekt mit den gleichen Eigenschaften und gleichem Namen am selben Ort erstellt und das Original gelöscht wurde.) Mit dem Befehl »Original finden« (⌘R) können Sie sich das Original zum Alias anzeigen lassen. Damit bei Aliasen ohne Type- und Creator-Kennungen das Symbol nicht verloren geht, schreibt Mac OS X zusätzlich eine »icns«-Ressource mit dem Symbol des Originals in die Alias-Datei.

In UNIX-Dateisystemen verweisen »Symbolic Links« auf andere Dateien. Diese erfüllen die gleich Aufgabe wie Aliase im HFS, sie besitzen jedoch andere Eigenschaften. Symbolic Links können in Mac OS X ebenfalls aufgelöst werden. Der Mac-OS-X-Finder kann jedoch keine Symbolic Links erzeugen; er erzeugt statt dessen ein Alias. Symbolic Links können im Programm »Terminal« in der Shell mit dem Befehl »ln -s« erzeugt werden.

Zusätzlich wird in UNIX-Dateisystemen mit Hardlinks gearbeitet. Mit Hardlinks ist es möglich, dass eine Datei sozusagen mehrere Namen hat. Hardlinks werden im Mac-Dateisystem in I-Node-Dateien verwaltet, die sich im unsichtbaren Ordner » HFS+ Private Data« auf der obersten Ebene der Festplatte befinden.

98 ▸
Praxis, Alias

53 ◂
i-Node



Der »X-Y Position Indicator for a Display System«, dessen Patentierung mit dieser Patentschrift 1967 von Doug Engelbart beantragt wurde, ist heute als »Maus« Standard an jedem Personal Computer.

Die Benutzeroberfläche des Xerox Alto von 1974 lässt nur erahnen, wohin die Entwicklung gehen würde. Das GUI wurde über drei Maustasten bedient.

Grafische Benutzeroberfläche

Der Gesichtssinn ist der ausgeprägteste der menschlichen Sinne. Die Grenzen der Sprache erfahren Sie recht schnell, beispielsweise bei dem Versuch, jemandem einen Gegenstand genau zu beschreiben, den Sie nicht sehen. So ist es z.B. einfacher, durch Zeigen auf eine Besonderheit hinzuweisen, als sie mit Worten zu erklären. Ähnlich ist es bei der Arbeit am Computer. Seine visuelle Orientierung macht es für einen Menschen einfach, einen Ordner zu aktivieren, indem er per Mausklick darauf zeigt. Er kann sich dann umsehen, ob sich das gesuchte Objekt – vielleicht eine Datei oder ein anderer Ordner – in diesem Ordner befindet. (Das Prinzip wird in Apples »Human Interface Guidelines« treffend »See and Point« genannt.) Mit Worten, die der Computer auch noch verstehen muss, den Weg zu einem Objekt zu beschreiben, ist da doch bedeutend komplizierter.

Eine erste Idee zu einer grafischen Benutzeroberfläche präsentierte Doug Engelbart schon im Jahre 1968 am Stanford Research Institute. Im Xerox-Entwicklungszentrum PARC wurde 1973 der Alto, ein Prototyp eines Rechners mit grafischer Benutzeroberfläche, gezeigt. Apple ergänzte diese Ideen um wesentliche Elemente wie z.B. Pull-Down-Menüs und Drag&Drop. Damit war der Standard gesetzt, seitdem gab es keine umgreifenden Änderungen mehr. Außerdem wurde die Maus weiterentwickelt. Die ursprünglich verwendete Dreitastenmaus wurde aufgrund der Erfahrungen bei Xerox zur Eintastenmaus, noch heute ein herausragendes Merkmal des Macs.

Diese grafische Benutzerschnittstelle war das im wahrsten Sinne des Wortes augenfälligste Merkmal des Mac. Vor dem Mac gab es keinen erschwinglichen Personalcomputer mit dieser Eigenschaft. In den darauf folgenden Jahren hat sich die grafische Benutzerschnittstelle auf allen Computerplattformen durchgesetzt.

Die Besonderheit des Mac-Systems war, dass diese Oberfläche ein integraler Bestandteil des Systems ist. In Mac OS X dagegen gibt es eine Trennung des eigentlichen Systems von der grafischen Oberfläche. Programme können über die Befehlszeile des Terminal-Programms

► 255ff
Praxis, Shell

angesteuert werden, BSD-Programme laufen sogar komplett ohne grafische Oberfläche. Echte Mac-OS-X-Programme (Carbon oder Cocoa) sind jedoch trotzdem ohne grafische Benutzeroberfläche nicht vorstellbar. In Mac OS X ist die Integration der grafischen Oberfläche so konsequent durchgezogen worden, dass selbst grundlegende Änderungen des Systems über diese Oberfläche vorgenommen werden können – durch Bewegen von Objekten oder Anklicken von Schaltflächen. Der normale Anwender kommt eigentlich nie mit dem Mac OS X zugrunde liegenden UNIX und dessen Shell in Kontakt.

Der Xerox Star, der erste kommerzielle Computer mit grafischer Benutzerberfläche erschien 1981, zwei Jahre Jahre vor der Lisa (jedoch nach einem Besuch von Apple-Mitarbeitern bei Xerox).

Der Finder

Der für den Anwender wichtigste Programm in Mac OS X ist der Finder. Der Finder liefert dem Anwender den Schreibtisch und die Volumes mit den Ordnerfenstern und den Dokumentensymbolen. Er bietet dem Anwender die Möglichkeit, in den Hierarchien der Festplatten zu navigieren und diese zu manipulieren. Der Finder bildet also die Schnittstelle zwischen dem Dateisystem und dem Anwender.

Der Finder gibt dem Anwender die Möglichkeit, seine Dateien übersichtlich zu ordnen. Wie in den »Human Interface Guidelines« auch für andere Programme empfohlen, verwendet der Finder dafür eine Metapher aus dem normalen Leben: Den (Büro-)Schreibtisch. Alle Dokumente lagern in Aktenschränken (den Volumes) in Ordnern. Um Einblick in einen Ordner zu bekommen, wird dieser aus dem Aktenschrank herausgenommen und man blättert in seinem Inhalt. Um ein bestimmtes Dokument zu betrachten, wird es aus dem Ordner herausgenommen und auf den Tisch gelegt. Ein Dokument, das nicht mehr gebraucht wird, landet im Papierkorb.

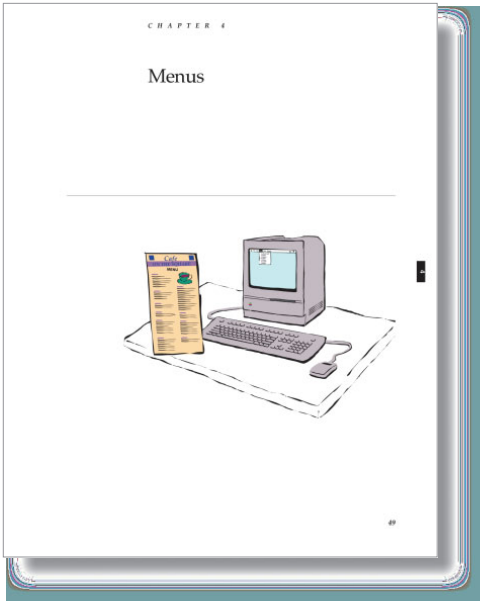
Zwar befindet sich der Papierkorb normalerweise nicht über dem Schreibtisch und wir werfen auch nicht einen ganzen Aktenschrank in den Papierkorb. Aber kleine Abweichungen von der Realität dienen der Konsistenz, wenn sie das System einfacher machen. Gleiche Handlungen erzeugen denselben Effekt.

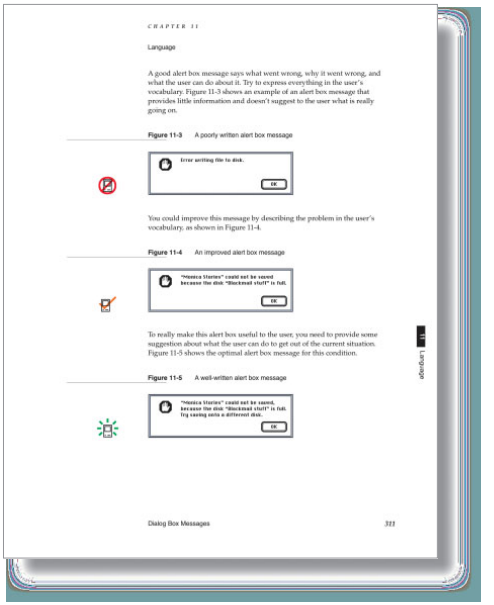
88 ▸
Praxis, Finder

Fenster-Darstellungs-Informationen

Damit der Finder die Fenster der einzelnen Ordner in der vom Benutzer gewünschten Form öffnen kann, muss er die Informationen zu den Darstellungen innerhalb der Fenster – z.B. Position und Größe der Icons, Größe des Fensters, Darstellungsart etc. – merken. Die Informationen werden innerhalb des jeweiligen Ordners in der unsichtbaren Datei ».DS_Store« gespeichert. Da die ».DS_Store«-Datei sich in dem zu öffnenden Ordner befindet und somit auch mitkopiert wird, kann auch beispielsweise für ein Diskimage oder ein Servervolume eine bestimmte Darstellung vorgegebenen werden.

Die Darstellungen des Schreibtischs und im Fenster »Computer« werden in der Voreinstellungsdatei des Finders gespeichert.





Human Interface Guidelines

Schon von Anfang an hat Apple detaillierte Richtlinien herausgegeben, wie ein Programm auf dem Mac auszusehen hat. Dafür wurden intensive ergonomische Studien durchgeführt. Aus dem Ergebnis dieser Studien entstanden einige grundlegende Prinzipien.

- **Direct Manipulation:** Der Anwender manipuliert immer das Objekt selbst und sieht direkt die Auswirkungen.

  Beispiel: Beim Verschieben eines Objekts bleibt dieses Objekt sichtbar. Das Objekt, auf das es gezogen wird, wird aktiviert.
- **See-and-Point:** Der Anwender bekommt immer alle Möglichkeiten präsentiert und kann dann zwischen diesen auswählen.

  Beispiel: Menüs präsentieren Befehle, unter denen dann ausgewählt wird.
- **Consistency:** Wenn zwei Objekte das gleiche tun, sehen sie auch gleich aus. Reagieren sie unterschiedlich, sehen sie auch unterschiedlich aus. So kann der Anwender schon aus dem Aussehen eines Objekts schließen, wie es sich verhalten wird.

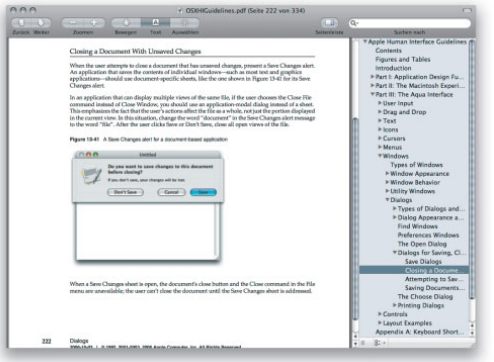
  Beispiel: Programme und Erweiterungsbundles unterscheiden sich dadurch, dass Programme vom Anwender gestartet werden, Erweiterungen jedoch vom System. Deshalb haben sie auch unterschiedliche Symbole.
- **User Control:** Der Anwender, nicht der Computer, kontrolliert den Verlauf einer Aktion.

  Beispiel: Die Hilfe erklärt dem Anwender lediglich, wie eine Einstellung am Objekt vorgenommen wird. Das nächste Mal kann der Anwender diese oder eine ähnliche Einstellung (Consistency) ohne Hilfe ausführen. Er ist nicht auf Gedeih und Verderb einem Assistenten ausgeliefert.
- **Forgiveness:** Der Anwender muss einen Schritt, den er ausgeführt hat, wieder rückgängig machen können.

Neben den hier aufgeführten Prinzipien gibt es noch viele weitere.

In den »Human Interface Guidelines« werden aber auch ganz konkrete Angaben zu Aussehen und Funktion vieler Objekte gemacht.

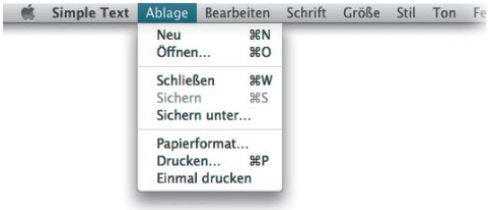
Die für »Aqua« überarbeiteten »Human Interface Guidelines« werden mit den Developer Tools als PDF- und als HTML-Dokument auf Ihrer Festplatte installiert (»/Developer/Documentation/UserExperience/Conceptual/AquaHIGuidelines«).

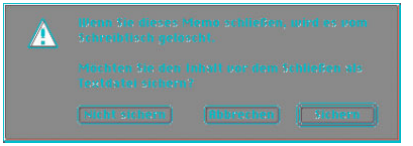


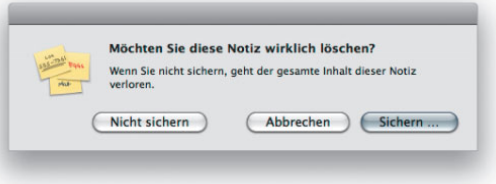
Beispiele aus den Interface Guidelines

- **Menüs:** Die beiden ersten Menüs »Ablage« und »Bearbeiten« sind als Standard vorgegeben, ebenso wie die Reihenfolge der Standardbefehle innerhalb dieser Menüs. Auch die Tastaturkürzel für die Grundfunktionen sind standardisiert. Diese Vorgaben wurden unverändert vom klassischen Mac OS in Mac OS X übernommen.
- **Abfragen:** Auch für die Sicherheitsabfrage zu einem geänderten Dokument gibt es Vorgaben. Da das Auge des Betrachters von oben links nach unten rechts wandert, sieht es zuerst das Symbol und liest dann den Text. Im Text muss zu erkennen sein, um welches Dokument es sich handelt. Schaltflächen, mit denen die getane Arbeit vernichtet wird (»Nicht sichern«), werden von den Schaltflächen getrennt, die die getane Arbeit nicht zerstören (»Abbrechen« und »Sichern«) – »Sichern« ist unten rechts und aktiviert. Die Beschriftung der Schaltflächen beschreibt noch einmal die Aktion – »Sichern« und »Nicht sichern», statt »Ja« und »Nein«.

  Unverständlicherweise finden sich immer wieder Programmhersteller, die sich nicht an diese ergonomischen Vorgaben halten.







Willkommen!

Der Systemstart

◂32 Firmware

Open Firmware...

Wenn Sie die Einschalttaste des PowerPC-Macs drücken, wird zuerst der »Power-On Self Test« (POST) aktiviert. Dieser überprüft die Schnittstellen, Erweiterungssteckplätze, Laufwerke und den Speicher. Wenn alles in Ordnung ist, erklingt der Startgong. Anschließend wird die Open Firmware aktiviert. Diese baut den Device-Tree auf – ein Verzeichnis aller installierten Hardware-Komponenten.

...bzw. BIOS und EFI

Beim Mac mit Intel-Architektur wird zuerst das BIOS aktiviert, das einen POST durchführt. Dann initialisiert das BIOS die grundlegende Hardware und der Prozessor wird in den 32-bit-Modus versetzt um das EFI laden zu können. Ist das EFI geladen, initialisiert es weitere Hardware und lässt den Startgong erklingen.

BootX bzw. boot.efi

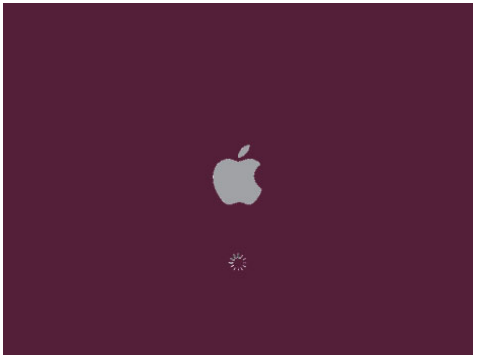
◂34 NVRAM
▸368 Referenz, Kontrollfeld, Startvolume

Die OpenFirmware bzw. EFI aktiviert die im NVRAM verzeichnete Boot-Datei. Wenn Mac OS X ausgewählt wurde, ist dies beim PowerPC-Mac die Datei »BootX« bzw. beim Mac mit Intel-Architektur »boot.efi« im Ordner »/System/Library/CoreServices«. Der graue Apfel erscheint auf dem Bildschirm.

◂68 Kernel-Environment

Kernel-Environment

BootX bzw. boot.efi lädt nun das Kernel-Environment. Dafür werden – beim PowerPC mithilfe des DeviceTrees – Treiber geladen, bis die Systemfestplatte aktiviert werden kann. Dann wird in verschiedenen Stufen der Mach-Kernel initialisiert. Beim Mac mit Intel-Architektur baut dieser außerdem den »fakePPCDevicetree« auf und aktiviert, wenn der Intel-Prozessor eine 64-bit-Erweiterung besitzt, in mehreren Schritten den 64-bit-Modus. (Ab hier ist der Ablauf bei PowerPC und Intel gleich.) Anschließend wird das BSD-System initialisiert. Unter dem Apfel wird ein Kreisel angezeigt.



Launchd

Anschließend wird der Daemon (ein unsichtbarer Systemdienst) »launchd« geladen. Der Bildschirm wechselt die Farbe und wird blau. Launchd startet verschiedene andere Systemdienste und ist auch für den Start aller weiteren Programme zuständig.

Unter den Diensten, die hier gestartet und anhand der Vorgaben in »/System/Library/LaunchDaemons« konfiguriert werden, befinden sich unter anderem AppleShare, Samba und der Apache-Webserver. Ab hier ist also ein Zugriff aus dem Netzwerk möglich.

188 ▶
Praxis, File-Sharing

Loginwindow

Das letzte Objekt, das geladen wird, ist das Programm »Loginwindow«. Dieses meldet den Benutzer beim System an. Das Programm bleibt immer aktiv, es ist auch für das Fenster »Sofort beenden« oder die Abmeldung zuständig.

292 ▶
Referenz, Sofort beenden

Benutzer-Programme

Loginwindow startet das Dock und den Finder mit dem Schreibtisch. Außerdem werden die im Kontrollfeld »Benutzer« unter »Startobjekte« bestimmten Objekte gestartet. Jetzt ist Mac OS X für die Arbeit bereit.

332 ▶
Referenz, Kontrollfeld »Benutzer«, »Anmeldeoptionen«