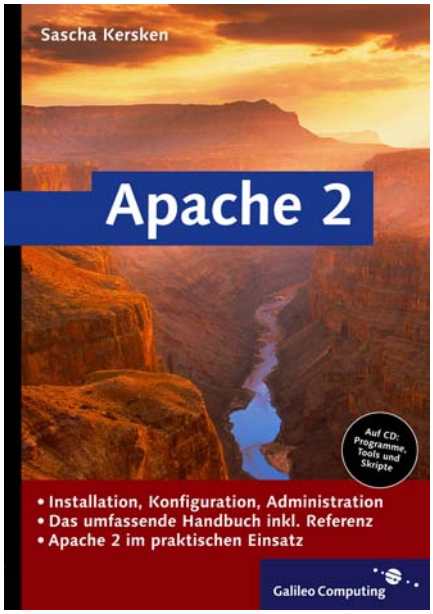

Sascha Kersken
Apache 2
Auf CD: Programme, Tools und Skripte
• Installation, Konfiguration, Administration
• Das umfassende Handbuch inkl. Referenz
• Apache 2 im praktischen Einsatz
Galileo Computing

# Auf einen Blick

# Inhalt

# Vorwort

*I love deadlines, especially the whooshing sound they make as they go by.*
*– Douglas Adams*

## Herzlich willkommen auf meiner Website!

Nein, natürlich ist dies keine Website, sondern unverkennbar ein Buch. Und doch hat der obige Satz viel mit dem Thema dieses Buches zu tun: In über 60% der Fälle, in denen Sie ihn lesen, wird er Ihnen von einem Apache-Webserver präsentiert. Damit ist Apache der mit Abstand verbreitetste Webserver und das erfolgreichste Open-Source-Softwareprojekt überhaupt. Seine Stabilität, Leistungsfähigkeit und Erweiterbarkeit lassen keine Wünsche offen.

Dies ist ein umfassendes Handbuch zum Apache-Webserver in der aktuellen Version 2.0; sämtliche Bestandteile, die zum »Lieferumfang« gehören, werden ausführlich beschrieben. Im Gegensatz zu den vielen Hybrid-Büchern, die bisher auf dem Markt sind, habe ich die Behandlung der Vorgängerversion 1.3 auf einen kurzen Anhang reduziert. Das macht das Buch übersichtlicher, weil nicht jeder zweite Satz ein Versionsvergleich ist. Apache 2 ist eine vollständige Neufassung des Programms; freie Software aus dem UNIX-Bereich würde ansonsten auch keine neue Hauptversionsnummer erhalten. Zwar haben die Entwickler versucht, die Konfiguration weitgehend kompatibel zu halten, aber immer gelingt dies natürlich nicht – besonders, weil Apache 2 über viel mehr Features verfügt als die älteren Versionen.

Die Schwerpunktthemen dieses Buches sind Installation, Administration und Programmierung. Sie erfahren zunächst einmal, wie Sie den Server unter verschiedenen Betriebssystemen kompilieren und / oder installieren können. Im weiteren Verlauf des Buches geht es vor allem um die unzähligen Konfigurationsanweisungen (Direktiven), die in der Hauptkonfigurationsdatei von Apache zur Verfügung stehen. Anders als viele andere Serverprodukte ist Apache nämlich von Hause aus nicht mit einer grafischen Konfigurationsoberfläche ausgestattet. Dies macht seine Administration zwar schwieriger, bietet aber dafür die größtmögliche Flexibilität.

Apache ist für zahlreiche verschiedene Plattformen und Betriebssysteme verfügbar. In der neuen Version wurde insbesondere die Unterstützung für Nicht-UNIX-Systeme verbessert: Als Basis der eigentlichen Server-Implementierung wurde eine Bibliothek namens Apache Portable Runtime (APR) eingeführt, die

statt der früher eingesetzten POSIX-Emulation die jeweiligen Stärken der einzelnen Systeme abstrahiert. Auch das Laufzeitverhalten wurde verbessert: Sie können nun aus mehreren so genannten Multiprocessing-Modulen (MPMs) das passendste für Ihre Plattform auswählen.

Ausführlich wird hier die Apache-Konfiguration für die Betriebssysteme UNIX (alle Varianten) und Windows (NT und seine Nachfolger) behandelt. Besonderheiten für andere Plattformen werden gegebenenfalls angemerkt, aber nicht weiter vertieft.

Einen großen Raum nehmen in diesem Buch die zahlreichen Module ein, die mit Apache 2 geliefert werden und für beinahe jeden Verwendungszweck eine praktische Lösung bieten. Auf diese Weise brauchen Sie bestimmte Aspekte der Funktionalität nur dann in Ihren Webserver zu integrieren, wenn Sie sie wirklich benötigen. Dies kann Ihnen helfen, den Überblick zu behalten und schont obendrein die Ressourcen des Server-Rechners.

**Das Komplettpaket**

Die beiliegende CD-ROM enthält so gut wie alle Programme, Listings und Dokumente, die in diesem Buch angesprochen werden[1]. Unter anderem finden sie darauf Apache-Distributionen für die verschiedensten Betriebssysteme, externe Module, Zusatzprogramme, Skripte und RFC-Dokumente. Die Tatsache, dass die Dateien auf diese Weise weiterverbreitet werden dürfen, ist einer der großen Vorteile freier Software.

Dennoch sollten Sie vor der Installation eines bestimmten Programms von der CD überprüfen, ob nicht bereits eine aktuellere Version verfügbar ist – bei Open-Source-Produkten bedeutet die Bereitstellung einer neuen Release oft, dass wichtige Sicherheitsprobleme aus vorangegangenen Versionen behoben wurden. Eine jeweils aktuelle Liste mit Download-Links und zahlreiche Zusatzinformationen finden Sie auf der Website zum Buch. Die Adresse lautet **buecher.lingoworld.de/apache2**.

## Das Buch im Überblick

In den einzelnen Kapiteln dieses Buches werden folgende Themen behandelt:

- In Kapitel 1, *Internet, WWW und IP-Netzwerke*, finden Sie eine kurze Übersicht über die Umgebung, in der Apache ausgeführt wird: In knapper Form

1 Einige wenige Tools dürfen aus rechtlichen Gründen nicht auf der CD verbreitet werden. In diesen Fällen finden Sie im Buch entsprechende Download-Links.

wird die Technik der TCP/IP-Protokollfamilie erläutert. Darüber hinaus gibt es hier auch eine Einführung in die Einrichtung eines Nameservers und Informationen über einige Hilfsprogramme.

- Kapitel 2, *Funktionsweise von Webservern*, behandelt das Anwendungsprotokoll HTTP, das die Grundlage des World Wide Web bildet. Neben der Besprechung sämtlicher HTTP-Methoden, -Header und -Statuscodes wird hier zur Veranschaulichung die Programmierung eines kleinen Webservers in Perl gezeigt.
- In Kapitel 3, *Apache 2 im Überblick*, wird in allgemeiner Form der Funktionsumfang des Webservers beschrieben. Dazu gehören auch Themen wie die Geschichte von Apache, ein Vergleich zu anderen Webservern sowie eine Liste der verfügbaren Module.
- Wie Sie Apache 2.0 auf Ihrem System installieren können, wird ausführlich in Kapitel 4, *Apache kompilieren und installieren*, beschrieben. Sie erhalten Anleitungen zur Kompilierung der Quellcode-Pakete unter UNIX und Windows sowie zur Installation diverser Binär-Distributionen.
- Kapitel 5, *Apache in Betrieb nehmen*, behandelt die Steuerung von Apache. Sie erfahren alles über das Starten, Stoppen und Neustarten des Servers sowie über Möglichkeiten, ihn beim Hochfahren des Systems automatisch zu starten.
- In Kapitel 6, *Grundkonfiguration*, wird zunächst der allgemeine Aufbau der Konfigurationsdatei `httpd.conf` erläutert. Anschließend werden alle Konfigurationsdirektiven besprochen, die für den Betrieb einer einfachen statischen Website wichtig sind.
- Kapitel 7, *Header und MIME-Einstellungen*, bietet weitere wichtige Informationen für die Administration von Websites: Apache 2 enthält Module und Konfigurationseinstellungen zur Manipulation von HTTP-Headern und MIME-Informationen. Dazu gehört auch das Thema Content-Negotiation, also die Belieferung von Clients mit deren jeweils bevorzugter Darstellungsform eines Dokuments.
- Das Thema von Kapitel 8, *Weiterleitung und Indizes*, sind Situationen, in denen sich unter der angeforderten URL kein Dokument befindet: URLs lassen sich auf Dokumente außerhalb des Website-Verzeichnisses oder sogar auf externe URLs umleiten; Apache kann zudem selbstständig Verzeichnisindizes generieren.
- In Kapitel 9, *Authentifizierung und gesicherte Verbindungen*, werden zwei wichtige Verfahren zur Absicherung von Websites behandelt: Die Authentifizierung, also die persönliche Anmeldung einzelner User, sowie die Bereitstellung SSL-gesicherter Verbindungen, die durch Verschlüsselung und

andere Maßnahmen vor Mitlese- oder gar Manipulationsversuchen geschützt werden.

- Kapitel 10, *Logging*, beschäftigt sich mit der Einrichtung und Verarbeitung von Logdateien. Diese wichtigen Helfer geben über alle Zugriffe auf Ihre Websites sowie über mögliche Fehler oder Angriffsversuche Aufschluss.
- Kapitel 11, *Skalierung, Load-Balancing und Proxies*, behandelt die wichtigsten Themen, die für den Betrieb besonders großer Websites relevant sind: Sie erfahren alles über den Einsatz von Apache als Proxy-Server, über virtuelle Hosts, Performance-Tuning und über Load-Balancing-Verfahren.
- In Kapitel 12, *CGI*, wird die klassische Schnittstelle zur Entwicklung von Web-Anwendungen vorgestellt, das Common Gateway Interface (CGI). Die Beispiele sind in Perl geschrieben; in diesem Zusammenhang lernen Sie das Perl-Modul `CGI.pm` kennen, das die Entwicklung von CGI-Skripten erheblich einfacher und komfortabler macht.
- Kapitel 13, *Technologien zur Web-Programmierung*, versammelt die beliebtesten Schnittstellen für die Entwicklung von Web-Anwendungen: PHP, `mod_perl` und Tomcat. Der Schwerpunkt ist die Integration der Module in Apache; darüber hinaus gibt es einige Programmierbeispiele und -tipps.
- In Kapitel 14, *SSI und Filter*, wird das interessante Konzept der Filter vorgestellt, das in Apache 2 neu eingeführt wurde: Eingehende Anfragedaten lassen sich ebenso leicht modifizieren wie die eigentlich schon fertige Antwort an die Clients. Das klassischs SSI-Verfahren (Server Side Includes) wurde in das Filterkonzept integriert und wird deshalb ebenfalls in diesem Kapitel behandelt.
- Kapitel 15, *Weitere Features*, behandelt zusätzliche Konzepte von Apache 2. Zunächst gibt es einen Überblick über Sicherheitsthemen: Als erstes eine Zusammenfassung der wichtigsten Hinweise aus den anderen Kapiteln; anschließend einige weitergehende Tipps. Den Rest des Kapitels nimmt die Beschreibung zusätzlicher Module ein. Unter anderem werden WebDAV, ISAPI und die Multiprotokoll-Unterstützung behandelt.

In den anschließenden Anhängen finden Sie zusätzliches interessantes Material: Eine kurze Übersicht über die Besonderheiten der Vorgängerversion Apache 1.3, einige Tabellen mit MIME-Types, Sprachkürzeln und Zeichensätzen, eine Zusammenfassung rechtlicher Aspekte und andere Themen.

## Danksagungen

Wieder einmal danke ich dem Team von Galileo Press, allen voran meinem Lektor Stephan Mattescheck. Er weiß am besten, warum das Vorwort gerade dieses Motto hat, und hat dennoch die logistische Meisterleistung vollbracht, das Buch noch rechtzeitig zu veröffentlichen. Darüber hinaus haben seine konstruktiven Anmerkungen immer wieder dazu beigetragen, dieses Buch noch besser zu machen.

Weiterer Dank gebührt natürlich den Entwicklern des Apache-Webservers. Ohne die zahllosen Stunden, die diese Enthusiasten freiwillig in die Entwicklung und Verbesserung dieses großartigen Produkts gesteckt haben, gäbe es das Thema dieses Buches gar nicht. In diesem Zusammenhang ist wohl auch eine Entschuldigung angebracht: Schon vor Monaten habe ich zugesagt, Gutachten zu deutschen Übersetzungen der Apache-Originaldokumentation anzufertigen. Wegen der Arbeit an diesem Buch bin ich bisher aber leider kaum dazu gekommen. Ich hoffe, dass das ab jetzt besser wird.

Einige Unternehmen haben mich für dieses Buch mit Software und Informationen versorgt. Ich danke vor allem der SuSE AG für die Bereitstellung von SuSE Linux 9.0 Professional und VMWare für die Workstation 4.0, ohne die das Schreiben unter Windows und die gleichzeitige Arbeit mit Apache auf diversen UNIX-Systemen undenkbar gewesen wäre.

Da ich im vorigen Buch vergessen habe, bestimmte Familienmitglieder zu erwähnen, mache ich es mir diesmal einfacher und danke der gesamten weit verzweigten Familie für jede Form der Unterstützung. Aber natürlich muss ich wie immer meine Frau Tülay und meinen Sohn Leon erwähnen, die mir auf jede erdenkliche Art behilflich waren und es sogar in der schwierigen Endspurtphase noch mit mir ausgehalten haben.

# 1 IP-Netzwerke, Internet und WWW

# 1 IP-Netzwerke, Internet und WWW

*Wer lange sinnt, beginnt nicht, und wer nicht beginnt, gewinnt nicht.*
*– Arabisches Sprichwort*

Der Apache-HTTP-Server (Kurzform »Apache«) ist ein Webserver für das Internet und das Intranet. Er dient vor allem dazu, bestehende oder dynamisch generierte Dokumente über ein Netzwerk an die Webbrowser von Benutzern auszuliefern. Internet und Intranet sind Bezeichnungen für globale beziehungsweise lokale Computernetzwerke, die die Netzwerkprotokollfamilie TCP/IP verwenden. Falls Sie mit all diesen Begriffen bereits vertraut sind, können Sie im nächsten Kapitel weiterlesen. Ansonsten finden Sie hier eine kurze Übersicht über TCP/IP-Netzwerke und das Internet. Eine viel ausführlichere Darstellung dieser Themen finden Sie beispielsweise in [HUNT2003][1].

Weiter unten in diesem Kapitel finden Sie noch eine Einführung in das Domain Name System (DNS) und die Konfiguration des Nameservers BIND. Diese Kenntnisse benötigen Sie, um einen öffentlich verfügbaren Webserver zu betreiben. Abgerundet wird dieses Kapitel schließlich durch die kurze Vorstellung einiger praktischer TCP/IP-Tools.

## 1.1 TCP/IP

Der Vorläufer des Internets, das so genannte ARPANet, wurde 1969 in Betrieb genommen. Aber erst knapp zehn Jahre später wurden die heutigen Netzwerkprotokolle eingeführt. Ein Netzwerkprotokoll ist ein Standard, der einen bestimmten Aspekt der Datenübertragung über ein Netzwerk regelt. »TCP/IP« ist der Name für eine ziemlich große Familie solcher Protokolle, benannt nach zwei ihrer wichtigsten Mitglieder: dem Transmission Control Protocol (TCP) und dem Internet Protocol (IP).

### 1.1.1 Das Internet-Schichtenmodell

Wie bei allen Computernetzen lassen sich auch beim TCP/IP-Netzwerk verschiedene Funktionsebenen oder Schichten unterscheiden. Diese Schichten betreffen jeweils eine andere Facette des Netzwerks.

Betrachten Sie für einen anschaulichen Vergleich das vorliegende Buch und seine »Schichten«:

1 Literaturangaben wie [HUNT2003] (Craig Hunt, »TCP/IP-Netzwerk-Administration«) werden in Anhang H aufgelöst.

1. Das Buch besteht aus bedrucktem Papier als »Trägermedium«.
2. Es enthält Linien, Kurven und Punkte, die von jemandem, der die lateinische Schrift lesen kann, als Buchstaben interpretiert werden.
3. Die Buchstaben ergeben Sätze, die jeder versteht, der der deutschen Sprache mächtig ist.
4. Die Sätze verbinden sich schließlich für diejenigen zu sinnvollen Informationen, die grundlegende IT-Kenntnisse haben und etwas über den Apache-Webserver wissen möchten.

Dass diese Ebenen voneinander getrennt betrachtet werden können, zeigt sich daran, dass jede für sich austauschbar ist, ohne die anderen zu beeinflussen:

1. Der Text muss nicht auf Papier gedruckt sein: Ich schreibe ihn an einem Rechner und sehe ihn auf dem Monitor; er könnte beispielsweise auch mit Zuckerguss auf eine (hinreichend große) Torte gespritzt oder in Stein gemeißelt werden.
2. Zur lateinischen Schrift gibt es bekanntermaßen zahlreiche Alternativen. Natürlich ist nicht jede Schrift gleich gut zur Wiedergabe jeder Sprache geeignet, aber prinzipiell kann man die Schrift wechseln und die Sprache beibehalten. Zum Beispiel wurde Türkisch früher mit arabischen Buchstaben geschrieben, heute dagegen mit leicht angepassten lateinischen (was übrigens besser funktioniert).
3. Wenn dieses Buch (lizenziert) in eine andere Sprache übersetzt würde, hätte ich überhaupt nichts dagegen ;-).
4. Nicht alle Leser benötigen ein Apache-Buch. Vielleicht möchten Sie zuvor ein UNIX-Grundlagenbuch wie zum Beispiel [WILL2002] lesen, oder Sie interessieren sich für ganz andere Themen und lesen lieber den neuen Harry Potter, einen Krimi oder ein Pflanzenbestimmungsbuch.

Mit Netzwerken verhält es sich so ähnlich. Das bekannte OSI-Referenzmodell – auf das hier nicht näher eingegangen wird – besitzt sieben solcher Schichten; zur Beschreibung des Internets und sonstiger TCP/IP-Netze reichen dagegen vier aus. Nach dem DDN Standard Protocol Handbook sind diese vier Schichten folgende:

1. Die **Netzzugangsschicht** beschreibt, auf welche Weise die angeschlossenen Rechner auf das Übertragungsmedium zugreifen, sich über die Sendereihenfolge einigen und mit Datenkollisionen umgehen.
2. Die **Internetschicht** beschreibt die eindeutige Adressierung der Rechner und die Weiterleitung von Daten über mehrere miteinander verbundene Netze hinweg, das so genannte **Routing**.

3. Auf der **Host-zu-Host-Transportschicht** werden die Daten in kleine Einheiten unterteilt, die Datenpakete, und von einer Anwendung auf dem einen Rechner (Host) zu einer Anwendung auf dem anderen geschickt.
4. Die **Anwendungsschicht** schließlich definiert, wie sich verschiedene Anwendungsprogramme über das Netzwerk »unterhalten«.

Der Ablauf der Netzwerkkommunikation ist immer gleich: Eine Anwendung auf einem Rechner – die sich natürlich auf der Anwendungsschicht befindet – übergibt Daten an den passenden Dienst der Host-zu-Host-Transportschicht. Dieser teilt die Daten in kleinere Einheiten ein, die so genannten Datenpakete, und versieht jedes Paket mit einer speziellen Transportinformation, dem Transport-Header. Diese vorbereiteten Pakete werden wiederum an die Internetschicht hinuntergereicht, die einen weiteren Header hinzufügt. Dieser enthält unter anderem die Absender- und die Empfängeradresse. Die Pakete, die mit all diesen zusätzlichen Informationen versehen wurden, heißen Datagramme. Erst in dieser Form werden sie tatsächlich der Netzwerkhardware anvertraut.

Der Empfängerrechner verfährt nun umgekehrt: Nachdem eine Funktion der Internetschicht festgestellt hat, dass die Daten tatsächlich für diesen Host bestimmt sind, reicht sie die Datagramme an den jeweils angesprochenen Transportdienst weiter. Dieser erkennt aus den Header-Informationen, für welche Anwendung die Daten bestimmt sind, und übergibt sie an diese.

Netzzugangsarten für TCP/IP-Netzwerke gibt es wie Sand am Meer; da sich die Internetprotokolle als der einzige ernst zu nehmende Standard für größere heterogene Netze durchgesetzt haben, kann TCP/IP mit beinahe jeder Art von Netzwerk- und DFÜ-Hardware unter so gut wie allen Betriebssystemen verwendet werden. Aus diesem Grund würde es den Rahmen dieser kurzen Übersicht bei weitem sprengen, sich hier mit einzelnen Zugangsformen zu befassen.

Ähnlich sieht es übrigens mit der obersten Schicht aus, der Anwendungsschicht. In diesem Buch geht es verständlicherweise vor allem um das Anwendungsprotokoll HTTP, das HyperText Transport Protocol, über das Webserver wie Apache und Webbrowser sich untereinander verständigen. Da Apache sich mit Hilfe optionaler Module auch für andere Server-Dienste wie FTP oder Mail verwenden lässt, werden Sie auch einige andere Protokolle der Anwendungsschicht kennen lernen. Da Anwendungsprotokolle jeweils sehr speziell sind und sich nicht verallgemeinern lassen, gehören auch sie nicht in dieses Kapitel.

Die beiden folgenden Unterabschnitte konzentrieren sich daher auf die beiden mittleren Schichten, auf denen die beiden Protokolle arbeiten, die der ganzen Protokollfamilie den Namen gegeben haben: die Internetschicht mit dem Internet Protocol (IP) und die Host-zu-Host-Transportschicht mit dem Transmission

Control Protocol (TCP) und seinem Konkurrenten, dem User Datagram Protocol (UDP).

### 1.1.2 Das Internet Protocol (IP)

Das Internet Protocol, die Monopollösung für die Internetschicht, löst vor allem zwei wichtige Aufgaben:

1. eindeutige Adressierung der Rechner und Netze,
2. Weiterleitung der Datenpakete vom Absenderrechner an den Empfänger, auch über mehrere Netze hinweg. Letzteres wird als Routing bezeichnet.

Zur Adressierung besitzt jeder Rechner im TCP/IP-Netz eine so genannte IP-Adresse. Es gibt zwei verschiedene Varianten des IP, die sich vor allem durch die Länge (und damit die verfügbare Anzahl) der Adressen unterscheiden. Das klassische IPv4-Protokoll verwendet 32 Bit lange Adressen, während die neuen IPv6-Adressen eine Länge von 128 Bit besitzen. IPv6 beginnt erst allmählich, sich durchzusetzen; der weit verbreitete Standard ist nach wie vor IPv4. Zwar wird dessen Adressraum im Internet allmählich knapp, aber der flächendeckende Umstieg auf die neue Version scheitert bisher an der Inkompatibilität der beiden Varianten zueinander sowie an der mangelnden IPv6-Unterstützung durch manche Netzwerkhard- und -software. Dennoch wird in diesem Buch auch die spezielle Konfiguration von Apache 2 für IPv6 erwähnt.

Wie alle Internetprotokolle und -standards ist auch das IP öffentlich verfügbar in so genannten RFC-Dokumenten (Request For Comments) beschrieben. Alle Personen, Institutionen und Firmen, die an der Weiterentwicklung des Internets beteiligt sind, stellen ihre Vorschläge und Lösungen seit über 30 Jahren in solchen freien Dokumenten zur Verfügung; es gibt inzwischen fast 3 600 davon. Sie können sie zum Beispiel unter **http://www.faqs.org/rfcs** lesen. IPv4 ist in RFC 791 dokumentiert; die neue Variante IPv6 in RFC 2460.

#### IPv4-Adressierung

Die 32 Bit langen IPv4-Adressen werden üblicherweise dezimal als einzelne, durch Punkte getrennte Bytes geschrieben, zum Beispiel **137.51.8.41**. Ein Teil dieser Adresse bezeichnet das Netzwerk, in dem sich der Rechner befindet, der Rest den Rechner selbst innerhalb dieses Netzes. Da es verschieden große Netzwerke gibt, haben die Entwickler von Anfang an verschiedene Arten – so genannte Klassen – von IPv4-Netzen geplant. Die Bits am Anfang der Adresse geben darüber Auskunft, zu welcher Klasse eine IP-Adresse gehört, und damit, wie viele Bits der Adresse das Netz und wie viele den Rechner (Host) kennzeichnen:

- Wenn das erste Bit 0 ist, gehört die Adresse zur Klasse A. Das erste der vier Bytes liegt also im Bereich 0 bis 127. In dieser Klasse kennzeichnen die ersten 8 Bits das Netz und die restlichen 24 den Host. Ein Klasse-A-Netz bietet jeweils 16,7 Millionen Host-Adressen.
- Wenn die ersten beiden Bits 10 sind, handelt es sich um die Klasse B; das erste Byte hat einen Wert zwischen 128 und 191. Hier stehen die ersten 16 Bit für das Netz und die anderen 16 für den Host. Ein Klasse-B-Netz enthält demnach 65.536 Host-Adressen.
- Sind die ersten drei Bits 110, dann gehört das Netz zur Klasse C. Das erste Byte hat also einen Wert zwischen 192 und 223. 24 Bits adressieren das Netz, und nur acht sind für den Host zuständig, so dass es nur 256 Host-Adressen in einem Netz gibt.
- Adressen, bei denen die ersten vier Bits 1110 sind und deren erstes Byte damit einen Wert zwischen 224 und 239 hat, gehören zur Klasse D. Sie dienen nicht der Adressierung einzelner Rechner, sondern sind so genannte Multicast-Adressen. Sie ermöglichen die Verteilung von Daten an mehrere Rechner, denen dieselbe Multicast-Adresse zugewiesen wird. Dies ist beispielsweise ideal für Videokonferenzen, Video-on-Demand oder andere Streaming-Dienste.

In Tabelle 1.1 sehen Sie eine genaue Übersicht über die IP-Adressklassen. Die Anzahl der Netzwerk-Bits ist doppelt angegeben: Der erste Wert gibt die Gesamtzahl an, während der Wert in Klammern die relevante Anzahl von Netz-Bits abzüglich der Klassenkennzeichnung angibt – bei der Klasse A beispielsweise 8 (7), da das erste Bit 1 lauten *muss*. Die Anzahl der pro Netz verfügbaren Hosts ist um zwei geringer, als sich rechnerisch ergeben. Das liegt daran, dass die niedrigste Adresse eines Netzes das Netzwerk selbst kennzeichnet und die höchste – die Broadcast-Adresse – dazu dient, innerhalb des Netzes Daten an alle angeschlossenen Rechner zu versenden.

| Klasse | Start-Byte | Netzwerk-Bits | Host-Bits | Anzahl Netze | Anzahl Hosts |
|---|---|---|---|---|---|
| A | 0-127 | 8 (7) | 24 | 128 | 16.777.214 |
| B | 128-191 | 16 (14) | 16 | 16.384 | 65.534 |
| C | 192-223 | 24 (21) | 14 | 2.097.152 | 254 |
| D | 224-239 | Multicast-Adressen | | | |

**Tabelle 1.1** Die IP-Adressklassen und ihre Eigenschaften

Jede IP-Adresse muss im Internet einmalig sein. Allerdings hat die IANA (Internet Assigned Numbers Authority) einige Adressen zur Verwendung in privaten, nicht direkt mit dem Internet verbundenen Netzwerken freigegeben:

- das Klasse-A-Netz 10.0.0.0,
- die 16 Klasse-B-Netze 172.16.0.0 bis 172.31.0.0,
- die 256 Klasse-C-Netze 192.168.0.0 bis 192.168.255.0.

Eine besondere Bedeutung haben darüber hinaus die folgenden IPv4-Adressen:

- 127.0.0.1 ist die so genannte Loopback-Adresse, über die ein Host mit sich selbst TCP/IP-Kommunikation betreiben kann. Beim Testen der Apache-Konfiguration und serverseitiger Anwendungen werden Sie diese Adresse oft verwenden, um Client und Server auf demselben Rechner zu betreiben.
- 255.255.255.255 ist die allgemeine Broadcast-Adresse. Sie kann zum Beispiel von einem Host verwendet werden, der beim Booten seine eigene IP-Adresse im Netz erfragt, weil sie über den Dienst DHCP dynamisch zugewiesen wird.
- Das Klasse-B-Netz 169.254.0.0 wird für APIPA (Automatic Private IP Addressing) oder »link local« verwendet – über diesen Dienst kann sich ein DHCP-Client selbst eine Adresse zuweisen, wenn aus irgendeinem Grund kein DHCP-Server verfügbar ist.

Als das Internet Protocol entwickelt wurde, schien die Anzahl der verfügbaren Netze, die sich aus der Klassenlogik ergibt, locker auszureichen. Da die Anzahl der Rechner und Netze im Internet jedoch schon seit langem exponentiell zunimmt, erweist sich das Klassensystem allmählich als zu starr und verschwenderisch: Allein die Hälfte aller rechnerisch verfügbaren Adressen wird für die Klasse A benutzt, obwohl selbst die größten Netze der Welt wohl kaum aus 16,7 Millionen Hosts bestehen. Aus diesem Grund wurde ein flexibleres System namens CIDR (Classless Inter-Domain Routing) eingeführt. Es ermöglicht das Setzen der Grenze zwischen Netz- und Host-Teil der Adresse bei einem beliebigen Bit. Bei einer CIDR-Adresse muss diese Grenze deshalb angegeben werden. Dafür gibt es zwei verschiedene Schreibweisen:

- Die Anzahl der Netzwerk-Bits wird durch einen Slash getrennt hinter die Adresse geschrieben: 192.78.16.97/24 ist beispielsweise die CIDR-Angabe für die Klasse-C-Adresse 192.78.16.97.
- Alternativ kann die Grenze zwischen Netz- und Host-Bits separat durch eine so genannte Subnet Mask (deutsch Teilnetzmaske) angegeben werden: Die Maske wird wie die IP-Adresse selbst dezimal in vier Einzel-Bytes geschrieben; Bits der Netzwerkadresse werden auf den Wert 1 gesetzt, Bits der Host-Adresse auf 0. Die Maske für ein Klasse-C-Netz ist demnach 255.255.255.0.

Die Besonderheit von CIDR wird deutlicher, wenn man eine Adresse betrachtet, die nicht zu einer der alten Klassen gehört, sondern eine individuelle Bit-Grenze besitzt: 160.76.153.12/19 gehört nicht zu dem Klasse-B-Netz 160.76.0.0 (CIDR-Adresse 160.76.0.0/16), sondern zu dem Netz 160.76.128.0/19. Es handelt sich um eines der Netze, die durch die Aufteilung des B-Netzes 160.76.0.0 in acht Teilnetze entstehen. Diese Netze werden in Tabelle 1.2 gezeigt. Mit einem Host-Teil von 13 Bit enthält jedes dieser Netze 8.190 Hosts (8.192 Einzeladressen inklusive Netzwerk- und Broadcast-Adresse). Die Subnet Mask lautet 255.255.224.0.

| Netzwerk | Erster Host | Letzter Host | Broadcast-Adresse |
|---|---|---|---|
| 160.76.0.0/19 | 160.76.0.1 | 160.76.31.254 | 160.76.31.255 |
| 160.76.32.0/19 | 160.76.32.1 | 160.76.63.254 | 160.76.63.255 |
| 160.76.64.0/19 | 160.76.64.1 | 160.76.95.254 | 160.76.95.255 |
| 160.76.96.0/19 | 160.76.96.1 | 160.76.127.254 | 160.76.127.255 |
| 160.76.128.0/19 | 160.76.128.1 | 160.76.159.254 | 160.76.159.255 |
| 160.76.160.0/19 | 160.76.160.1 | 160.76.191.254 | 160.76.191.255 |
| 160.76.192.0/19 | 160.76.192.1 | 160.76.223.254 | 160.76.223.255 |
| 160.76.224.0/19 | 160.76.224.1 | 160.76.255.254 | 160.76.255.255 |

**Tabelle 1.2** Alle Netze, die durch CIDR-Aufteilung (Subnetting) des Klasse-B-Netzes 160.76.0.0 entstehen

Noch flexibler als CIDR ist die so genannte VLSM-Adressierung (Variable Length Subnet Mask). Durch dieses Adressierungsschema lässt sich ein Netzwerk in Teilnetze unterschiedlicher Größe unterteilen, was beispielsweise in einer Firma mit unterschiedlich großen Abteilungen sehr nützlich ist.

Ein weiterer nützlicher IP-Dienst, der übrigens die Knappheit der IPv4-Adressen um einige Jahre verzögern konnte, ist NAT (Network Address Translation). Ein NAT-Gateway ersetzt die privaten IP-Adressen (siehe oben) ausgehender Datenpakete durch öffentliche Adressen und verfährt bei eingehenden Datenpaketen umgekehrt. Auf diese Weise kann das Netzwerk mit den oben erläuterten privaten Adressen versehen werden und dennoch mit dem Internet verbunden sein. Zudem erhöht dieses Verfahren die Sicherheit, weil die eigentliche Infrastruktur des Netzes vor potenziellen Eindringlingen verborgen wird.

### IPv6-Adressierung

IPv6-Adressen sind, wie erwähnt, 128 Bit lang. Dies ergibt einen rechnerischen Adressraum von unvorstellbaren $3{,}4 \times 10^{38}$ Adressen. Auch wenn diese Anzahl

von Hosts wohl niemals erreicht werden wird,[2] besitzt eine so große Adressenzahl andere Vorteile: Erstens sind Unmengen von Netzen mit zahlreichen verschiedenen Größen möglich, und zweitens passt die IPv6-Adressierung zu den zahlreichen Geräten, die inzwischen neben den klassischen Allround-Computern ins Internet drängen: Handys, PDAs, Armbanduhren und sogar Kühlschränke und Mikrowellengeräte.

Da es etwas unhandlich wäre, die IPv6-Adressen in 16 dezimalen Byte-Gruppen zu schreiben, werden sie statt dessen in acht vierstelligen Hexadezimalblöcken dargestellt. Eine IPv6-Adresse lautet also beispielsweise so: 09AC:7B03:CC3D:BD88:0000:0037:6293:CCF1. Sie lässt sich abkürzen, indem die führenden Nullen jeder Gruppe weggelassen werden und indem eine zusammenhängende Gruppe aus reinen Nullenblöcken durch :: dargestellt wird. Die Beispieladresse lautet in dieser Schreibweise also folgendermaßen: 9AC:7B03:CC3D:BD88::37:6293:CCF1.

Die Grenze zwischen Netzwerk- und Host-Teil der IPv6-Adressen wurde von Anfang an flexibel gesetzt; wie bei IPv4-Adressen wird die Bit-Anzahl des Netzwerk-Teils durch einen / getrennt hinter die Adresse geschrieben; dabei kann der Rest der Adresse weggelassen werden, wenn das Netzwerk als solches gemeint ist, zum Beispiel 9AC:7B03/32.

Während die automatische Zuweisung der Adressen sich bei IPv4 nur durch nachträglich erfundene Dienste wie DHCP oder APIPA realisieren lässt, ist sie bei IPv6 die Regel. Aus diesem Grund wird der Host-Teil der Adresse bei einem Ethernet-LAN beispielsweise automatisch aus der 48 Bit langen MAC-Adresse (Media Acces Control; eindeutige Hardware-Adresse) der Netzwerkschnittstelle – ergänzt um einen 16 Bit langen Kennzeichnungsblock – gebildet.

In der Praxis wird IPv6 heute vornehmlich durch **Tunnelung** benutzt: Die IPv6-Datenpakete werden in IPv4-Datenpakete »verpackt« (erhalten also einen IPv4-Paket-Header) und über das IPv4-adressierte Internet zum Ziel geleitet – ein Netzwerk, das intern die entsprechende IPv6-Adressierung versteht. Es gibt zahlreiche kommerzielle und freie Anbieter solcher IPv6-Tunneldienste. Einer der bekanntesten ist das Projekt 6bone (**www.6bone.net**).

### IP-Routing

Das Wichtigste am Konzept des IPs ist die Fähigkeit, Daten über mehrere miteinander verbundene Netze weiterzuleiten. Diese Aufgabe übernehmen spezi-

2 Obwohl man mit einer solchen Einschätzung vorsichtig sein sollte – wer weiß, welche Möglichkeiten die entstehende Nanotechnik mit sich bringt ... Außerdem konnten sich auch die IPv4-Entwickler nicht im Traum vorstellen, dass der Adressraum jemals ausgeschöpft werden könnte. Und in den 1940er Jahren schätzte IBM-Präsident Tom Watson den Weltbedarf(!) an Computern auf fünf Stück.

alisierte Rechner, die Router. Sie sind jeweils mit mindestens zwei verschiedenen Netzwerken verbunden und verfügen über Informationen darüber, welche Datenpakete sie wohin weiterleiten müssen, damit diese letztlich ihr Ziel erreichen. Während diese Routing-Informationen innerhalb kleinerer Firmennetze statisch konfiguriert werden, führen die Router der Internet-Provider und anderer Betreiber großer Netze so genannte Routing-Protokolle aus, mit deren Hilfe die Routing-Konfiguration automatisch veröffentlicht und verbreitet wird.

Auch jeder Host im TCP/IP-Netzwerk benötigt grundlegende Routing-Informationen, um Daten jeweils über die richtige Netzwerkschnittstelle an den korrekten Router weiterzuleiten. Aus der Sicht eines Hosts gibt es zwei Sorten von Routern: Einen »normalen« Router, an den der Host Daten weitergibt, deren Ziel ein bestimmtes Netzwerk ist, und das Default Gateway (der Standard-Router), dem alle Datenpakete übergeben werden, für deren Ziel eben kein spezieller Router konfiguriert wurde.

Angenommen, das Netzwerk 196.23.17.0/24 einer kleinen Firma mit der Domain **mynet.de** ist über den Router **intern** (196.23.17.2) mit dem internen Nachbar-Teilnetz 196.23.18.0/24 verbunden, während **extern** (196.23.17.1) das Default Gateway ist. Im Teilnetz 196.23.18.0/24 besitzt **intern** die IP-Adresse 196.23.18.1 und ist dessen einziger Router und damit Default Gateway.

Aus diesem Szenario ergibt sich für einen Host im Netz 196.23.17.0/24 die folgende Routing-Situation:

- Datenpakete für Rechner im Netzwerk 196.23.17.0/24 werden ohne Umweg über einen Router direkt an den Empfänger versandt.
- Daten, die für Hosts im Netz 196.23.18.0/24 bestimmt sind, werden dem Router **intern** (196.23.17.2) zur Weiterleitung übergeben.
- Daten für alle anderen Netze, also für das gesamte Internet, werden an den Router **extern** (196.23.17.1) weitergegeben.

Ein Host im Netzwerk 196.23.18.0/24 besitzt dagegen eine einfachere Routing-Konfiguration; da es in diesem Teilnetz nur einen Router gibt, brauchen nur zwei Fälle unterschieden zu werden:

- Daten für das Netz 196.23.18.0/24 werden ohne Routing unmittelbar zugestellt.
- Die restlichen Datenpakete werden dem Router **intern** (hier: 196.23.18.1) übergeben; dieser weiß natürlich, dass er Daten für das Netz 196.23.17.0/24 über seine Schnittstelle 196.23.17.2 unmittelbar zustellen kann, während er alle anderen an den nächsten Router, **extern** (196.23.17.1) weiterleiten muss.

Damit ein IP-Datagramm nicht endlos im Netz kreist, wenn sein Bestimmungsort nicht gefunden werden kann, enthält sein Header ein 8 Bit breites Feld namens TTL (Time To Live). Jeder einzelne Router, den das Paket passiert, zieht von diesem Wert 1 ab. Sobald er 0 geworden ist, verwirft der Router das Paket. Ein Datagramm kommt also entweder innerhalb von 256 Hops (Routing-Schritten) an oder gar nicht. In der Regel sind allerdings erheblich weniger Hops erforderlich.

### 1.1.3 Transportprotokolle

Aufgabe der Host-zu-Host-Transportschicht ist die direkte Lieferung der Datenpakete von einer Anwendung auf einem Host zu einer Anwendung auf einem anderen Host. Ein Transportprotokoll nimmt die Daten von einem Anwendungsprogramm entgegen, das über das Netzwerk kommunizieren möchte. Es teilt die Daten in Datenpakete auf, versieht sie mit einer eindeutigen Nummer, die die entsprechende Anwendung kennzeichnet, und reicht sie an das IP weiter, das die tatsächliche Datenweiterleitung über das Netzwerk durchführt.

Im Internet-Protokollstapel gibt es zwei verschiedene Transportprotokolle. Jede Netzwerkanwendung kann sich dasjenige aussuchen, das besser zu ihr passt: Das TCP garantiert die Auslieferung jedes einzelnen Datenpakets und garantiert durch eine Datenpaketnummerierung, dass die Pakete am Ziel wieder in der richtigen Reihenfolge zusammengesetzt werden können. Das UDP ist dagegen erheblich schneller als TCP, kann aber nur einzelne Pakete ohne Reihenfolge übertragen und überprüft auch nicht, ob sie tatsächlich ankommen. Während TCP also datenstromorientiert ist, ist UDP nachrichtenorientiert.

#### TCP

Das TCP (RFC 793) sorgt für die Übertragung eines kontinuierlichen Datenstroms. Obwohl netzwerktypisch noch immer jedes einzelne Paket einen beliebigen Routing-Weg zum Ziel wählen kann, wird durch die eingebaute Flusskontrolle eine virtuelle Punkt-zu-Punkt-Verbindung zwischen den beiden Hosts eingerichtet.

Die TCP-Verbindung beginnt durch den so genannten Drei-Wege-Handshake:

- Der Host, der die Verbindung initiiert (üblicherweise der Client, denn er »will etwas« vom Server), sendet dem Empfänger ein Datenpaket, in dessen TCP-Header ein spezielles Bit namens SYN gesetzt ist.
- Der Empfänger antwortet mit einem Paket, in dem die beiden Flags SYN und ACK gesetzt sind.

- Der ursprüngliche Absender beantwortet dieses Paket noch einmal mit einem Datenpaket, in dem nur ACK gesetzt ist.

Diese Vorgehensweise garantiert, dass beide Seiten sowohl senden als auch empfangen können. Die Flusskontrolle kommt dadurch zustande, dass der Header jedes Pakets eine Sequenznummer enthält. Sie gibt den Offset des Datenbytes für den Teil des Gesamtdatenstroms an, der mit diesem Paket versandt wird. Der Absender erwartet dann für jedes einzelne Paket eine Bestätigung, in der das ACK-Bit gesetzt ist und die eine Bestätigungsnummer enthält: den Byte-Offset des nächsten erwarteten Pakets. Falls eine Bestätigung innerhalb einer bestimmten Zeit ausbleibt, sendet der Absender das Paket einfach erneut.

Die Identifikation der beteiligten Anwendungen erfolgt durch ein Paar 16 Bit langer Portnummern. Da eine bestimmte Verbindung durch beide Portnummern gekennzeichnet wird, ist es möglich, dass ein TCP-Server jeweils eine feste Portnummer hat, während ein Client eine zufällige Nummer (»Ephemeral Ports«) erhält. Die wichtigsten Server-Dienste haben festgelegte Ports (»Well-Known Ports«) zwischen 0 und 1.023 – ein Webserver hat zum Beispiel standardmäßig die Portnummer 80. Clients dagegen erhalten Portnummern ab 1.024 bis 65.535.

### UDP

Wenn Einzeldaten ihr Ziel so schnell wie möglich erreichen sollen, es aber nicht auf Vollständigkeit oder auf die richtige Reihenfolge einer größeren Datenmenge ankommt, kann eine Anwendung statt TCP das UDP verwenden. Dieses Protokoll ist in RFC 768 spezifiziert. Der Name hat damit zu tun, dass die IP-Datagramme der Anwendung ohne viel zusätzlichen Aufwand zur Verfügung stehen.

Der Hauptgrund, warum UDP schneller ist als TCP, liegt am wesentlich kleineren Paket-Header. Das Verhältnis zwischen Konfigurations- und Nutzdaten ist also erheblich günstiger als beim TCP. Der Header enthält nämlich nichts weiter als die beiden bei TCP beschriebenen Portnummern (hier Service-Nummern genannt), Paketlänge und Prüfsumme.

Ein UDP-Datenpaket ist seiner Natur gemäß eine einzelne Nachricht, die so schnell wie möglich ans Ziel befördert wird. Der Absender kümmert sich nicht darum, ob sie ankommt oder nicht, und erwartet auch keine Empfangsbestätigung. Man könnte TCP gewissermaßen als »Einschreiben mit Rückschein« bezeichnen, während UDP eher einer Postkarte entspricht.

## 1.2 Das Domain Name System (DNS)

Für die Netzwerkkommunikation zwischen Computern sind IP-Adressen eine hervorragende Einrichtung, und zwar sowohl IPv4- als auch die längeren IPv6-Adressen. Menschen dagegen bevorzugen den Umgang mit Namen gegenüber der Verwendung von Nummern. Aus diesem Grund ist es nützlich, wenn die Hosts im Internet neben der numerischen Adresse auch einen für Menschen leichter zu merkenden Namen besitzen. Das Problem ist nur, dass dieser Name auf effiziente und wenig störanfällige Weise in die IP-Adresse umgewandelt werden muss.

Als das ARPANet in Betrieb genommen wurde, bestand es aus sehr wenigen großen Rechnern an verschiedenen amerikanischen Universitäten. Zwar wuchs es von Anfang an recht schnell, aber selbst mit einigen hundert beteiligten Knoten war es noch kein allzu großes Problem, eine Liste der Hostnamen mit den entsprechenden Adressen als Textdatei zu pflegen, weil sich Änderungen eher in Grenzen hielten. Diese Datei hieß `hosts.txt` und wurde regelmäßig an die beteiligten Institutionen verteilt.

Noch heute enthält jeder Computer mit einem Betriebssystem aus der UNIX-Familie eine Datei namens `/etc/hosts`, die zur lokalen Namensauflösung dient. Unter Windows befindet sie sich dagegen unter **%Systemroot%\System32\drivers\etc\hosts**[3]. Diese Datei besitzt folgende Struktur:

```
127.0.0.1       localhost
196.23.17.1     defaultgw   defaultgw.mynet.de
196.23.17.2     mailserver  mailserver.mynet.de
```

In jeder Zeile steht also eine IP-Adresse, gefolgt von einem oder mehreren Hostnamen, die in diese Adresse aufgelöst werden sollen. In den meisten Systemen lässt sich konfigurieren, ob Einträge in dieser Datei beim Nachschlagen von Namen Vorrang vor externen Nameservern haben sollen oder nicht.

Im Laufe der Jahre wuchs das Internet immer weiter, und irgendwann war der Austausch der stets aktualisierten **hosts.txt**-Datei nicht mehr effektiv genug. Aus diesem Grund machte man sich an die Arbeit, ein neues System mit einer verteilten Datenbank aufzubauen. Dieses System heißt **DNS** (Domain Name System); das Grundkonzept wird in RFC 1034 und 1035 beschrieben.

3 **%Systemroot%** ist eine Windows-Umgebungsvariable, die für das Systemverzeichnis steht – häufig ist es **C:\Windows**.

### 1.2.1 Das DNS-Konzept

DNS besteht im Wesentlichen aus zwei wichtigen Elementen: zum einen den Nameservern, die eine DNS-Server-Software ausführen und auf Anfrage Namensinformationen herausgeben, und zum anderen aus dem hierarchischen System der Domain-Namen selbst.

Das System der Domain-Namen ist in etwa mit einem Dateisystem vergleichbar, das Verzeichnisse, Unterverzeichnisse und Dateien enthält. Angenommen, ein UNIX-Dateisystem enthält die folgende Datei:

```
/home/sascha/books/apache2/kap01.txt
```

Dies bedeutet, dass sich die Datei `kap01.txt` im Verzeichnis `apache2` befindet. Dieses Verzeichnis ist ein Unterverzeichnis von `books`; dies wiederum ist ein Unterverzeichnis von `sascha` im Verzeichnis `home`. Das Verzeichnis `home` schließlich befindet sich unterhalb des Wurzelverzeichnisses `/`.

Bei einem DNS-Domain-Namen verhält es sich ähnlich, allerdings ist die Reihenfolge der Hierarchie umgekehrt. Betrachten Sie etwa den folgenden Domain-Namen:

```
www.buecher.lingoworld.de
```

Es handelt sich um den Host/Dienst `www` in der Subdomain `buecher`, die sich in der Second-Level-Domain `lingoworld` unterhalb der Top-Level-Domain (TLD) `de` – ISO-Länderkürzel für Deutschland – befindet. Die eigentliche Wurzel des DNS ist an dieser Stelle nicht sichtbar; ihr Name ist der leere String »«.

Wer seine Rechner im Internet unter einem Domain-Namen betreiben möchte, muss zunächst eine Second-Level-Domain unter der passenden Top-Level-Domain registrieren. Je nach TLD ist jeweils eine andere Registrierungsstelle zuständig. Um die Top-Level-Domain `de` kümmert sich beispielsweise die DENIC (**www.denic.de**). In der Praxis ist es jedoch einfacher und meist billiger, einen Domain-Namen über einen Provider zu registrieren.

Für typische kleine bis mittlere Unternehmen, die nichts weiter als eine Website und vielleicht ein paar E-Mail-Adressen benötigen, kommt hier am ehesten ein virtueller Host (siehe Kapitel 11, *Skalierung, Load-Balancing und Proxies*) in Frage: Ein Domain-Name wie **www.mynet.de** verweist auf eine Website, die sich in Wirklichkeit zusammen mit hunderten anderer Sites auf demselben Server-Rechner bei dem Provider befindet. Darüber hinaus lassen sich natürlich E-Mail-Adressen nach dem Schema **user@mynet.de** anlegen.

Wenn Sie dieses Buch aus praktischen Gründen lesen, ist ein virtueller Host bei einem großen Hosting-Dienst natürlich nicht das Richtige für Sie. Sie benötigen

Ihren eigenen Server-Rechner (oder auch mehrere), auf dem Sie Apache selbst nach Ihren Wünschen konfigurieren können. Möglicherweise wollen Sie auch selbst Hosting-Dienstleistungen für Ihre Kunden erbringen. Im ersten Fall wird sich ebenfalls der Provider um die DNS-Konfiguration kümmern; **mynet.de** verweist nun eben nicht mehr auf einen virtuellen, sondern auf einen tatsächlichen Host. Wenn Sie dagegen selbst Hoster sind, müssen Sie die virtuellen oder echten Hosts für Ihre Kunden einrichten, und dazu gehören natürlich auch die korrekten DNS-Einstellungen.

Eine DNS-Domain, die in der Obhut eines bestimmten Netzbetreibers steht, wird als **DNS-Zone** bezeichnet. Im Beispiel **www.buecher.lingoworld.de** ist **lingoworld.de** eine solche Zone. Der Administrator von **lingoworld.de** kann Hosts innerhalb dieser Zone einrichten, indem er DNS-Daten mit ihren Hostnamen und IP-Adressen erstellt. Sollte er eine Subdomain wie **buecher.lingoworld.de** anlegen, kann er die Hosts und sonstigen Informationen darin entweder selbst konfigurieren oder aber die Administration dieser Subdomain delegieren und somit zur neuen, unabhängigen Zone machen. Dieses System der optionalen Delegation garantiert größtmögliche Flexibilität.

Interessant ist schließlich, wie eine DNS-Anfrage funktioniert. Ein Host, der Daten an einen bestimmten Empfänger im Internet senden möchte, aber nur dessen Hostnamen kennt, muss einen Nameserver befragen, um die zugehörige IP-Adresse zu erhalten.

Jeder Host, der direkt mit dem Internet verbunden ist, kennt mindestens einen Nameserver – entweder wurde dieser bei der Netzwerkkonfiguration manuell angegeben, oder er wurde bei der automatischen Schnittstellenkonfiguration über DHCP (Dynamic Host Configuration Protocol; im LAN) oder PPP (Point-to-Point Protocol; bei Modem, ISDN, DSL) vom entsprechenden Anmelde-Server übermittelt.

Der Host, der eine Namensauskunft wünscht, befragt nun die ihm bekannten Nameserver der Reihe nach. Angenommen, es wird die IP-Adresse zu **www.othernet.com** gesucht. Ein Nameserver, der diese Anfrage erhält, versucht nun, diese Information zu ermitteln und an den Anfragenden zurückzugeben. Damit er dieselbe Antwort beim nächsten Mal schneller geben kann, speichert er die ermittelten Daten in einem Cache (Zwischenspeicher). Wenn der befragte Nameserver bereits den Nameserver kennt, der für die Zone **othernet.com** zuständig ist, wendet er sich direkt an diesen, nimmt die Antwort in Empfang und gibt sie aus. Andernfalls muss er »eine Etage höher« nachfragen: Die Root-Nameserver für die Top-Level-Domain `de` kennen auf jeden Fall mindestens einen Nameserver für jede unterhalb dieser TLD regist-

rierte Domain. Mit dieser Information kann der ursprünglich beauftragte Nameserver also weiterforschen und erhält schließlich die passende Antwort.

Es gibt im Grunde drei Arten von Nameservern:

- primäre Master-Nameserver,
- Slave-Nameserver,
- Caching-Only-Nameserver.

Auf einem primären Master-Nameserver werden die maßgeblichen – die so genannten **autoritativen** – Informationen für eine DNS-Zone konfiguriert. Dieser Server ist »der Weisheit letzter Schluss« für die entsprechende Zone.

Ein Slave-Nameserver dient der **Replikation**, enthält also eine Kopie der Daten eines primären Master-Nameservers. Dies dient sowohl der Ausfallsicherheit als auch der Entlastung des Master-Servers. Aus Sicherheitsgründen ist es ratsam, die DNS-Daten des eigenen primären Master-Nameservers auf mindestens einem Slave-Nameserver zu speichern, der sich nicht innerhalb der eigenen Netzwerkinfrastruktur befindet. Größere Unternehmen und Internet-Provider stellen einander diese Dienstleistung oft gegenseitig zur Verfügung.

Caching-Only-Nameserver werden oft in kleineren Unternehmen eingesetzt, die ihre öffentlichen Server nicht innerhalb ihrer Geschäftsräume betreiben. Diese Server enthalten keinerlei autoritative Informationen und sind auch keine direkten Replikationspartner primärer Master-Nameserver, sondern beschleunigen durch die Zwischenspeicherung lediglich wiederholte Zugriffe auf externe DNS-Daten.

### 1.2.2 Der DNS-Server BIND

Es gibt verschiedene Implementierungen von Server-Software, die den DNS-Dienst versieht. Die wichtigste von ihnen ist BIND (Berkeley Internet Naming Domain). Es handelt sich um Open-Source-Software des Internet Software Consortiums (ISC). Sie können BIND und entsprechende Informationen darüber von der Website **www.isc.org/products/BIND** und zahlreichen dort verzeichneten Mirror-Sites herunterladen. Die zurzeit aktuelle Version ist BIND 9.2.3. Beachten Sie jedoch, dass noch immer nicht alle Features von BIND 8 in die neue Release 9 integriert wurden – in einigen (inzwischen sehr seltenen) Fällen kann es vorkommen, dass Sie ein spezielles Problem lösen müssen, für das Sie Version 8 benötigen. Die kurze Übersicht in diesem Unterabschnitt bezieht sich auf BIND 9. Wenn Sie ausführlichere Informationen benötigen, finden Sie eine praxisorientierte Anleitung in [LIU2003].

Eine andere weit verbreitete Nameserver-Software ist der Microsoft DNS Server. Er ist in die Windows-Server-Betriebssysteme Windows 2000 Server und Windows Server 2003 integriert. Selbstverständlich erfüllt er nach außen dieselbe Funktion wie BIND, da er zum globalen DNS kompatibel ist. Da die Nutzung eines Windows-Servers in aller Regel mit der Verwendung des Microsoft Internet Information Servers (IIS) als Webserver einhergeht, braucht die Konfiguration dieses Dienstes in einem Buch über den Apache nicht beschrieben zu werden.

### BIND installieren

In aktuellen UNIX- oder Linux-Distributionen ist BIND normalerweise bereits enthalten. Sie können den Server also problemlos über den Paketmanager Ihres Systems installieren. Wenn Sie nicht genau wissen, wie das funktioniert, konsultieren Sie die Dokumentation Ihres Systems oder gehen Sie analog zu dem Verfahren vor, das für verschiedene Betriebssysteme in Kapitel 4, *Apache kompilieren und installieren*, für den Apache-Webserver beschrieben wird.

Wenn BIND nicht mit Ihrem System geliefert wird oder wenn Sie die neueste Version installieren möchten, sollten Sie BIND im Sourcecode herunterladen und selbst kompilieren. Das funktioniert, wie bei autoconf/automake-Software üblich, nach dem folgenden bewährten Schema:

- Entpacken Sie BIND in Ihr Source-Verzeichnis (eine inzwischen optionale Konvention; irgendein anderes Verzeichnis tut es auch):

  ```
  # cd /usr/src
  # tar -zxvf /tmp/bind-9.2.3.tar.gz
  ```

  (es folgen zahlreiche Ausgabezeilen)

  ```
  # cd bind-9.2.3
  ```

- Rufen Sie das Konfigurationsprogramm und anschließend `make` auf, um BIND mit den passenden Optionen für Ihr System zu kompilieren:

  ```
  # ./configure
  # make
  ```

- Rufen Sie zu guter Letzt den Installationsbefehl auf:

  ```
  # make install
  ```

Normalerweise wollen Sie anschließend dafür sorgen, dass der Nameserver – das Programm `named` – beim Booten automatisch gestartet wird. Da dies unter verschiedenen UNIX-Systemen (und selbst bei den verschiedenen Linux-Distri-

butionen) unterschiedlich funktioniert, können Sie sich auch hier an die Anleitung halten, die in Kapitel 5, *Apache in Betrieb nehmen,* für den automatischen Start von Apache für Ihr Betriebssystem zur Verfügung gestellt wird.

### BIND konfigurieren

Die Konfigurationsdaten von BIND bestehen aus zwei verschiedenen Dateien: Die Datei `/etc/named.conf` enthält die Konfigurationsinformationen über den Nameserver selbst, während so genannte Zonendaten-Dateien die eigentlichen Namensdaten in Form von **Resource Records** enthalten. Beide Dateiarten sind einfache Textdateien, die Sie mit Ihrem bevorzugten Editor[4] bearbeiten können.

Die erste wichtige Angabe in der Datei `named.conf` beschreibt das Arbeitsverzeichnis, in dem sich die Zonendaten-Dateien befinden. Da diese Anweisung global ist und keine bestimmte Zone betrifft, wird sie in einen `options`-Block gesetzt:

```
options {
   directory "/var/named";
};
```

Anschließend folgen ein oder mehrere `zone`-Blöcke. Sie enthalten die Konfiguration der Zonen, für die der Nameserver zuständig sein soll.

Falls Ihr Nameserver beispielsweise primärer Master-Nameserver für die Zone `mynet.de` sein soll, sieht der entsprechende `named.conf`-Eintrag so aus:

```
zone "mynet.de" {
   type master;
   file "db.mynet.de";
};
```

Der Dateiname `db.mynet.de` ist nicht vorgeschrieben. Sie können jeden beliebigen Namen wählen, aber `db.name-der-zone` ist üblich. Natürlich müssen Sie die entsprechende Datei auch anlegen.

Für jede Zone, für die Ihr Nameserver als primärer Master dient, benötigen Sie zusätzlich eine Reverse-Lookup-Zone. Dies ist eine Zone, die den umgekehrten Dienst leistet: die Umwandlung einer gegebenen IP-Adresse in einen Domain-Namen. Der Name dieser Zone hängt von der Größe des Subnets ab, in dem sie sich befindet: Es handelt sich um den umgekehrten Netzwerk-Teil der IP-Adresse mit dem reservierten Domain-Namen `in-addr.arpa`. Angenommen,

4 *Meiner* ist unter UNIX der GNU Emacs – aber das ist Geschmackssache.

mynet.de verwendet das Netz 196.17.23.0/24, dann wird die folgende Reverse-Lookup-Zone konfiguriert:

```
zone "17.23.196.in-addr-arpa" {
   type master;
   file "db.196.23.17";
};
```

Wenn Ihr Nameserver als Slave dienen soll, beispielsweise für die Zone somenet.de, sieht die entsprechende zone-Anweisung so aus:

```
zone "somenet.de" {
   type slave;
   masters { 138.19.47.3; };
   file "bak.somenet.de";
};
```

Unter masters wird eine Liste von Nameservern angegeben, von denen der Slave die Replikationsdaten erhält. Sie brauchen keine primären Master-Nameserver zu sein; der Begriff »masters« bezieht sich auf übergeordnete Nameserver aus der Perspektive des Slaves.

Schließlich benötigen Sie auf jeden Fall noch einen Eintrag für die Root-Hints-Datei. Diese spezielle Datei enthält Informationen über die Server, die den Stamm des DNS bilden und auf die Root-Nameserver der verschiedenen TLDs verweisen. Die entsprechende Datei (zum Beispiel named.root) ist entweder in Ihrer BIND-Distribution enthalten, oder Sie müssen sie per FTP unter der Adresse ftp.rs.internic.net/domain/named.root herunterladen. Der Eintrag in named.conf sieht so aus:

```
zone "." {
   type hint;
   file "named.root";
};
```

Bitte beachten Sie, dass die Syntax der Datei named.conf ganz exakt eingehalten werden muss – aus eigener Erfahrung weiß ich, dass Programmierer besonders gern das Semikolon hinter der schließenden geschweiften Klammer vergessen.

### Zonendaten

Falls Ihr Nameserver als primärer Master für eine Zone dient, müssen Sie nun die entsprechende Zonendaten-Datei anlegen. Sie enthält Resource Records für die einzelnen Hosts, Server-Dienste und andere Elemente der Zone. Die erste

Zeile einer Zonendaten-Datei ist eine `$TTL`-Anweisung (Time To Live) – sie gibt an, wie lange andere Nameserver die aus dieser Datei enthaltenen Informationen maximal im Cache halten dürfen. Sie können den Wert entweder komplett in Sekunden angeben oder mit den folgenden Maßeinheiten arbeiten: `w` (Wochen), `d` (Tage), `h` (Stunden), `m` (Minuten) oder `s` (Sekunden). Wenn der Wert 28 Stunden betragen soll, gibt es beispielsweise die folgenden drei Möglichkeiten:

```
$TTL 100800
$TTL 28h
$TTL 1d4h
```

Die nächste Information ist ein SOA-Record (Start of Authority). Er enthält die folgenden Konfigurationsinformationen über die Zone selbst:

- **MNAME**.
  Der Hostname des primären Master-Nameservers.
- **RNAME**
  Die E-Mail-Adresse des Verantwortlichen; das @ wird durch einen Punkt ersetzt.
- **Die Seriennummer der Zone**
  Sie sollte bei jeder Aktualisierung erhöht werden; ein praktisches Format für manuell gepflegte Zonendaten ist JJJJMMTTVV (Letzteres ist eine zweistellige Versionsnummer, die bei 00 beginnt und wichtig ist, wenn mehrere Änderungen an einem Tag stattfinden).
- **Der Refresh-Wert**
  Gibt das Intervall an, in dem die Slave-Nameserver anfragen sollen, ob die Zonendaten aktualisiert wurden.
- **Der Retry-Wert**
  Bestimmt, wie lange ein Slave warten soll, bevor er nach einem Verbindungsfehler erneut nach Aktualisierungen fragt.
- **Der Expire-Wert**
  Gibt an, wie lange die Slaves antworten sollen, wenn der primäre Master nicht erreichbar ist. Dieser Wert sollte recht hoch sein, weil er für Ausfallschutz sorgt.
- **Der Negative-Caching-Wert**
  Gibt an, wie lange die Slaves negative Antworten (»nicht gefunden« usw.) im Cache speichern dürfen. Der Wert sollte recht klein sein, da es sich um einen vorübergehenden Ausfall halten könnte.

Ein vollständiger SOA-Record für `mynet.de` sieht zum Beispiel so aus:

```
mynet.de.     IN  SOA  ns1.mynet.de.  (
              hostmaster.mynet.de.
              2003112501
              30m
              10m
              30d
              30m )
```

Die (runden!) Klammern sind nur dann erforderlich, wenn Daten der Übersicht halber auf mehrere Zeilen verteilt werden.

Wenn Sie einen Host zu einer Zonendaten-Datei hinzufügen möchten, benötigen Sie einen A-Record (Address). Für `pc1.mynet.de` mit der IP-Adresse 196.17.23.24 sieht ein solcher Eintrag so aus:

```
pc1.mynet.de.    IN    A    196.17.23.24
```

Außerdem benötigen Sie einen PTR-Record (Pointer) in der Zonendaten-Datei der Reverse-Lookup-Zone (in diesem Beispiel in `db.196.17.23`). Für `pc1.mynet.de` sieht dieser Eintrag so aus:

```
24.23.17.196.in-addr.arpa.    IN    PTR    pc1.mynet.de.
```

Auf diese Weise müssen Sie alle Rechner in Ihrer Domain, die öffentlich über Hostnamen verfügbar sein sollen, angeben – natürlich auch Ihre eigenen Nameserver, Webserver, Mailserver und so weiter.

Für Webserver-Betreiber ist es oft nützlich, wenn der reine Domain-Name (`mynet.de`) ebenso auf den Webserver verweist wie `www.mynet.de`; viele Benutzer versuchen den Zugriff ohne das Präfix `www`, weil sie das von großen Sites wie `google.com` gewohnt sind. Zu diesem Zweck brauchen Sie nur einen zusätzlichen A-Record einzurichten, der auf die IP-Adresse des Webservers verweist. Beide Möglichkeiten zusammen sehen also so aus:

```
www.mynet.de.    IN    A    196.23.17.3
mynet.de.        IN    A    196.23.17.3
```

Einen CNAME-Record (Canonical Name; ein Alias, der auf einen anderen Hostnamen verweist) dürfen Sie in diesem Zusammenhang nicht verwenden. Nützlich ist der CNAME-Record aber beispielsweise dann, wenn Ihr Webserver intern anders heißt als `www`:

```
www.mynet.de.    IN    CNAME    winnetou.mynet.de
```

Wenn das Ziel des Alias-Records (der Hostname auf der rechten Seite) sich in einer anderen Zone befindet als der Alias-Name, muss dieser Record in der Zone stehen, in die der Alias-Name gehört.

Sie können sogar über die Zonendaten bereits eine triviale Form von Load-Balancing betreiben. Wenn Sie mehrere Webserver betreiben, können Sie für jeden von ihnen einen A-Record schreiben:

```
www.mynet.de.    IN   A   196.23.17.3
www.mynet.de.    IN   A   196.23.17.4
www.mynet.de.    IN   A   196.23.17.5
```

Die IP-Adressen der Webserver werden dann im so genannten Round-Robin-Verfahren (reihum) ausgegeben. Professionelles Load-Balancing (siehe Kapitel 11, *Skalierung, Load-Balancing und Proxies*) benötigt allerdings eine komplexere Konfiguration, die nicht nur DNS betrifft, weil eine gerechte Verteilung der Webserver-Ressourcen beispielsweise auch von der Dateigröße der ausgelieferten Dokumente abhängt und nicht nur von der reinen Anzahl der Namensanfragen.

Ein weiterer wichtiger Typ von Resource Records sind die NS-Records (Nameserver). Sie geben sämtliche autoritativen Nameserver für die Zone an – wie bereits erwähnt, sollte mindestens einer dabei sein, der sich nicht innerhalb des eigenen Netzwerks befindet. Für die Domain `mynet.de` könnten die NS-Records also beispielsweise so aussehen:

```
mynet.de.    IN   NS   ns1.mynet.de.
mynet.de.    IN   NS   ns2.mynet.de.
mynet.de.    IN   NS   ns1.provider.de.
```

Mit Hilfe von NS-Records wird übrigens auch die Delegation untergeordneter Zonen durchgeführt: Für eine Subdomain wird einfach ein anderer Nameserver angegeben. Sofern dieser Nameserver sich innerhalb der Zone befindet, die in der aktuellen Zonendaten-Datei konfiguriert wird, muss zusätzlich ein A-Record für diesen Nameserver angegeben werden:

```
accounting.mynet.de.      IN   NS   ns.accounting.mynet.de.
accounting.mynet.de.      IN   NS   ns2.provider.de.
ns.accounting.mynet.de.   IN   A    196.23.17.9
```

Zu guter Letzt benötigen Sie noch MX-Records (Mail Exchange) für die Angabe von Mail-Zielen. Schließlich sollen Benutzer E-Mails an Adressen wie **user@mynet.de** schicken können statt an **user@mail.mynet.de**. MX-Records enthalten eine Prioritätsangabe, die festlegt, welcher Mailserver bevorzugt werden soll. Ein Server mit einem höheren Wert wird nur gewählt, wenn der-

jenige mit dem nächstkleineren offline oder aus anderen Gründen nicht verfügbar ist:

```
mynet.de.    IN    MX     0 mail.mynet.de.
mynet.de.    IN    MX    10 snailmail.mynet.de.
mynet.de.    IN    MX    20 mail.provider.de.
```

Es gibt noch zahlreiche andere Typen von Resource Records. Beispielsweise wurde hier nicht auf die recht komplexe DNS-Konfiguration für IPv6 eingegangen. Wenn Sie selbst für die Verwaltung Ihrer DNS-Zonen verantwortlich sind und öffentliche Nameserver betreiben müssen, kommen Sie nicht umhin, sich entsprechende Literatur zu beschaffen.

## 1.3 TCP/IP-Diagnose und -Fehlersuche

Beim Arbeiten mit TCP/IP-Netzwerken kommt es mitunter zu Ausfällen und sonstigen Problemen. Einige einfache Programme, die zur TCP/IP-Implementierung fast aller Betriebssysteme gehören, ermöglichen die Diagnose solcher Schwierigkeiten. In diesem Abschnitt werden die wichtigsten von ihnen kurz vorgestellt.

### 1.3.1 ping

Das einfachste, aber eines der wichtigsten TCP/IP-Dienstprogramme ist `ping`. Der Name ist kein Akronym, sondern ahmt das Geräusch eines Echolots nach. Genau diese Aufgabe erfüllt `ping` auch: An einen beliebigen Host gesendete Kontrolldatenpakete werden von diesem zurückgesendet. Wenn jedes Paket ordnungsgemäß zurückgesendet wird, ist die Verbindung gewährleistet. Zusätzlich gibt das Programm die Antwortgeschwindigkeit in Millisekunden aus.

Die Syntax von `ping` ist sehr einfach: Sie brauchen nur `ping hostname` beziehungsweise `ping ip-adresse` einzugeben. Das Programm sendet ein Datenpaket nach dem anderen an den Empfänger, bis Sie es mittels [STRG] + [C] abbrechen. Anschließend wird eine Statistik mit mittlerer Dauer und dem Prozentsatz verloren gegangener Pakete angezeigt.

Die Windows-Version von `ping` sendet standardmäßig nur vier Pakete. Hier müssen Sie die Option `-t` verwenden, um beliebig viele Datenpakete zu senden.

Die `ping`-Ausgabe sieht beispielsweise so aus:

```
$ ping www.heise.de
PING www.heise.de (193.99.144.71): 56 data bytes
```

```
64 bytes from 193.99.144.71: icmp_seq=0 ttl=255 time=0.172 ms
64 bytes from 193.99.144.71: icmp_seq=1 ttl=255 time=0.099 ms
64 bytes from 193.99.144.71: icmp_seq=2 ttl=255 time=0.095 ms
64 bytes from 193.99.144.71: icmp_seq=3 ttl=255 time=0.093 ms
64 bytes from 193.99.144.71: icmp_seq=4 ttl=255 time=0.094 ms
64 bytes from 193.99.144.71: icmp_seq=5 ttl=255 time=0.093 ms
64 bytes from 193.99.144.71: icmp_seq=6 ttl=255 time=0.093 ms
64 bytes from 193.99.144.71: icmp_seq=7 ttl=255 time=0.098 ms
-- www.heise.de ping statistics --
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 0.093/0.104/0.172 ms
```

`ping` verwendet übrigens kein TCP oder UDP, sondern ein eigenes, noch leichtgewichtigeres Protokoll namens ICMP (Internet Control Message Protocol). Es wird in RFC 792 beschrieben.

Beachten Sie, dass die Erreichbarkeit eines Hosts per `ping` nichts weiter bedeutet, als dass er prinzipiell mit dem Netzwerk beziehungsweise Internet verbunden ist. Dies sagt nichts über seine Aufgabe im Netzwerk oder über die Funktionsbereitschaft von Server-Diensten aus.

Dennoch ist `ping` ein sehr praktisches Hilfsmittel. Beispielsweise können Sie durch »Anpingen« verschiedener Hosts unterscheiden, welche Netzwerkschnittstelle oder welcher Router unter Umständen ausgefallen ist.

### 1.3.2 traceroute

Das Programm `traceroute` verfolgt automatisch die Route zum angegebenen Host über das Internet. Dazu setzt es den TTL-Wert seiner Testpakete nacheinander auf 1, 2, 3 und so weiter. Ein Router, bei dem das aktuelle Paket den Wert 0 erreicht, verwirft es aber nicht wie üblich automatisch, sondern sendet eine Meldung zurück. `traceroute` sortiert die Antworten nach dem ursprünglichen TTL-Wert und gibt so nacheinander die Router aus, die bis zum Erreichen des Ziels passiert werden. Das Ganze sieht zum Beispiel so aus:

```
$ traceroute www.heise.de
traceroute to www.heise.de (193.99.144.71), 30 hops max, 40 byte
 packets
1 erx-maw1.netcologne.de (195.14.247.95) 71.634 ms
2 swrt-maw1-g34.netcologne.de (213.196.239.169) 60.253 ms
3 cat6509-pg1-vl200.netcologne.de (195.14.195.145) 58.283 ms
4 rtint4-ge11.netcologne.de (81.173.192.2) 72.938 ms
5 rtdecix-g01.netcologne.de (81.173.192.85) 56.923 ms
```

```
6 de-cix2.ffm.plusline.net (80.81.193.132) 73.637 ms
7 c22.f.de.plusline.net (213.83.57.53) 64.342 ms
8 www.heise.de (193.99.144.71) 70.263 ms
```

`traceroute` ist beispielsweise dann nützlich, wenn `ping` keinen Erfolg gezeigt hat: Sie können erkennen, ob der Fehler in Ihrem eigenen Netzwerk oder »irgendwo da draußen« liegt.

Beachten Sie, dass die Windows-Version des Programms `tracert` heißt.

### 1.3.3 netstat

Das Programm `netstat` liefert zahlreiche Informationen über den Netzwerkzustand. Wenn Sie den Befehl ohne weitere Angaben eingeben, erhalten Sie eine Übersicht über sämtliche aktuellen Netzwerkverbindungen. Sie können diese Liste mit Hilfe einer Option einschränken, sinnvollerweise auf TCP. Unter Linux heißt die entsprechende Option `--tcp` oder `-t`, unter vielen anderen UNIX-Versionen sowie unter Windows `-p tcp`. Andere Angaben wie UDP sind nicht sonderlich sinnvoll, da es keine »UDP-Verbindungen« gibt – in der Praxis würden die Kanäle angezeigt, über die vor kurzem UDP-Datagramme versandt wurden.

Für TCP sieht die Ausgabe zum Beispiel so aus:

```
$ netstat -t
Active Internet connections (w/o servers)
Proto  Recv-Q  Send-Q  Local Address   Foreign Address State
Tcp    0   0   localhost:1032  localhost:1033  ESTABLISHED
Tcp    0   0   localhost:1033  localhost:1032  ESTABLISHED
Tcp    0   0   localhost:1030  localhost:1034  ESTABLISHED
Tcp    0   0   localhost:1031  localhost:1030  ESTABLISHED
Tcp    0   0   localhost:1028  localhost:1029  ESTABLISHED
Tcp    0   0   localhost:1029  localhost:1028  ESTABLISHED
Tcp    0   0   localhost:1026  localhost:1027  ESTABLISHED
Tcp    0   0   localhost:1027  localhost:1026  ESTABLISHED
Tcp    0   0   localhost:1024  localhost:1025  ESTABLISHED
Tcp    0   0   localhost:1025  localhost:1024  ESTABLISHED
```

Die Option `-r` erfüllt eine andere Option: Sie gibt die aktuellen Routing-Tabellen für alle Schnittstellen aus. Das sieht beispielsweise folgendermaßen aus:

```
$ netstat -r
Aktive Routen:
Netzwerk  Ziel            Netzmaske        Gateway       Schnittst. Metrik
          0.0.0.0         0.0.0.0  213.196.251.226  213.196.251.226      1
```

```
      127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
    192.168.0.0    255.255.255.0      192.168.0.2     192.168.0.2       2
    192.168.0.2  255.255.255.255        127.0.0.1       127.0.0.1       1
  192.168.0.255  255.255.255.255      192.168.0.2     192.168.0.2       1
```

### 1.3.4 nslookup

Das Programm `nslookup` befragt einen Nameserver nach der IP-Adresse zum angegebenen Hostnamen oder umgekehrt. Dabei muss die IP-Adresse des befragten Nameservers angegeben werden, sofern in der TCP/IP-Konfiguration des Betriebssystems keine festen Nameserver angegeben wurden.

Das Ergebnis sieht beispielsweise wie folgt aus, wenn der erste Nameserver von NetCologne (194.8.194.70) nach der IP-Adresse für `www.heise.de` befragt wird:

```
$ nslookup www.heise.de 194.8.194.70
Server:  ns1.netcologne.de
Address:  194.8.194.70

Nicht-autorisierte Antwort:
Name:    www.heise.de
Address:  193.99.144.71
```

Die umgekehrte Art der Abfrage sieht etwa so aus:

```
$ nslookup 193.99.144.71 194.8.194.70
Server:  ns1.netcologne.de
Address:  194.8.194.70

Name:    www.heise.de
Address:  193.99.144.71
```

Das Programm `dig` dient ebenfalls dem Befragen von Nameservern, gibt aber ausführlichere Antworten. Beispielsweise kann die Art von DNS-Resource-Records angegeben werden, nach denen gesucht wird. Mit `dig` lauten die oben gezeigte Suche nach der IP-Adresse (also einem A-Record) für den Host `www.galileocomputing.de` und ihre Ausgabe so:

```
# dig @194.8.194.70 a www.galileocomputing.de
; <<>> DiG 2.2 <<>> @194.8.194.70 a www.galileocomputing.de
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10
```

```
;; flags: qr aa rd ra; Ques: 1, Ans: 1, Auth: 2, Addit: 0
;; QUESTIONS:
;; www.galileocomputing.de, type = A, class = IN

;; ANSWERS:
www.galileocomputing.de.  86400   A   80.237.200.227

;; AUTHORITY RECORDS:
galileocomputing.de.      259200  NS  a1.rpns.hosteurope.de.
galileocomputing.de.      259200  NS  c1.rsns.hosteurope.de.

;; ...truncated
;; Total query time: 131 msec
;; FROM: prefect to SERVER: 194.8.194.70
;; WHEN: Wed Nov 26 12:52:22 2003
;; MSG SIZE  sent: 42  rcvd: 113
```

Übrigens wurde `dig` inzwischen auch für Windows portiert. Unter **bind8nt.meiway.com/resources.cfm** (der empfehlenswerten Website »BIND on Windows NT«) finden Sie den entsprechenden Download-Link.

### 1.3.5 telnet

Das Programm `telnet` beherrscht nicht etwa nur das gleichnamige Protokoll zur Terminalemulation, sondern ermöglicht auch die Kommunikation mit sämtlichen textbasierten Server-Diensten. Dabei können Sie die Eingaben, die normalerweise von einer Client-Software vorgenommen werden, manuell vornehmen. Auf diese Weise erhalten Sie einen direkten Einblick in Vorgänge, die normalerweise hinter den Kulissen stattfinden, und können so mögliche Kommunikationsfehler aufspüren.

Das Folgende ist beispielsweise der Mitschnitt einer Telnet-Session mit dem Webserver **www.apache.org**. Ob es sich dabei um einen Apache-Server handelt, ist in diesem Zusammenhang unerheblich. Jedenfalls entsprechen die manuellen Eingaben in `telnet` (die Zeilen von `GET / HTTP/1.1` bis `Connection: close` sowie eine zusätzliche Leerzeile) den Daten, die normalerweise Ihr Browser an den Webserver übermittelt. Die Ausgabe enthält nicht nur den HTML-Code, den ein Browser umsetzen würde (und der hier auf die erste Zeile mit der `<!DOCTYPE>`-Angabe beschränkt wurde), sondern davor zunächst den HTTP-Header. Die Bedeutung aller hier verwendeten Anfrage- und Antwort-Header wird in Kapitel 2, *Funktionsweise von Webservern*, genau erläutert.

```
$ telnet www.apache.org 80
Trying 209.237.227.195...
Connected to www.apache.org.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.apache.org
Connection: close

HTTP/1.1 200 OK
Date: Wed, 31 Mar 2004 09:06:48 GMT
Server: Apache/2.0.49-dev (Unix)
Last-Modified: Sun, 28 Mar 2004 22:42:31 GMT
ETag: "2da7807-267d-b64da3c0"
Accept-Ranges: bytes
Content-Length: 9853
Cache-Control: max-age=86400
Expires: Thu, 01 Apr 2004 09:06:48 GMT
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
               "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transiti-
onal.dtd">
[...]

Connection closed by foreign host.
```

Statt der Portnummer 80 hätten Sie auf den meisten Systemen auch einfach `http` schreiben können – die wichtigsten Well-Known-Ports haben Namen, die in der Datei **/etc/services** (Windows: **%Systemroot%\System32\Drivers\etc\services**) festgelegt sind. Eine noch vollständigere Liste erhalten Sie unter **www.iana.org/assignments/port-numbers**.

Für die Terminalemulation, für die `telnet` ursprünglich entwickelt wurde, sollten Sie das Programm übrigens nach Möglichkeit nicht einsetzen. Das Problem ist nämlich, dass alle Daten – einschließlich Passwörtern – im Klartext über das Netzwerk beziehungsweise Internet übertragen werden. Eine vernünftige Alternative ist SSH (Secure Shell). Dieses Protokoll verwendet eine recht sichere Verschlüsselung; allerdings können Sie nicht das Programm `telnet` dafür verwenden. Die meisten UNIX-Varianten enthalten den Kommandozeilen-Client `ssh`; für Windows ist beispielsweise PuTTY geeignet (kostenloser Download etwa unter **www.chiark.greenend.org.uk/~sgtatham/putty**).

# 3 Apache 2 im Überblick

# 3 Apache 2 im Überblick

*Wer etwas Wichtiges vorhat, sollte nicht lange Reden halten, sondern nach ein paar Worten zur Sache kommen.*
*– Sagoyewatha, Red-Jacket-Indianer*

Nach den diversen Vorabinformationen in den ersten beiden Kapiteln erhalten Sie hier einen Überblick über den Webserver Apache 2. Der erste Abschnitt enthält einen kurzen historischen Abriss über Apache und seine Vorläufer, über die Apache Software Foundation, die seine Weiterentwicklung koordiniert, sowie einen Vergleich mit einigen bekannten Konkurrenzprodukten. Im zweiten Abschnitt geht es dagegen technischer zu: Sie erhalten kurz gefasst die wichtigsten Informationen über die Funktionen und Eigenschaften des Servers.

## 3.1 Einführung

Der Apache-HTTP-Server oder Apache-Webserver ist ein Gemeinschaftsprodukt von zahlreichen Softwareentwicklern, die ihre Arbeit über öffentliche Mailing-Listen und verteilte CVS-Repositories[1] koordinieren. Es handelt sich um ein – wenn nicht sogar um *das* – Prestigeprodukt aus der Welt der freien Software oder Open-Source-Software[2]: Kaum ein anderes Programm aus dieser Szene ist so beliebt und verbreitet wie Apache. Selbst Microsoft-Gründer Bill Gates kann nicht anders, als es zuzugeben: »Apache, free software, has been used by large companies for a long time.« (c't-Interview, Ausgabe 4/1999).

Da die Quellcodes des Programms öffentlich verfügbar sind, kann jeder (ausreichende C-Kenntnisse vorausgesetzt) den Server genau auf seine eigenen Bedürfnisse zuschneiden. Besonders geglückte Lösungen für spezifische Probleme können Sie natürlich auch der Apache-Nutzergemeinschaft zukommen lassen.

Wie Sie im nächsten Kapitel sehen werden, können Sie aber sogar ohne Programmierkenntnisse von dieser Flexibilität profitieren: Wenn Sie den Server selbst kompilieren, können Sie zuvor zahlreiche Einstellungen vornehmen, um ihn an Ihre Betriebssystemumgebung und Ihre speziellen Wünsche anzupassen.

1 Zurzeit (März 2004) wird der Umstieg auf Subversion vorbereitet. Subversion ist ein Apache-Modul und baut auf das in Kapitel 15 besprochene WebDAV auf.

2 Der Begriff »Open-Source-Software« wurde 1998 als pragmatischere Bezeichnung für freie Software geprägt. In diesem Buch wird bevorzugt der ursprüngliche Begriff »freie Software« verwendet. Das Wort »frei« steht übrigens eher für uneingeschränkt verwendbar als für kostenlos.

### 3.1.1 Entstehungsgeschichte des Apache-Webservers

Der erste Webserver überhaupt wurde als Referenzimplementierung der WWW-Idee von Tim Berners-Lee und seinem Team am europäischen Kernforschungszentrum geschrieben. Aus diesem Grund wurde das Programm später zur Unterscheidung von anderen Webservern als CERN httpd bezeichnet – »httpd« steht für **HyperText Transport Protocol Daemon**.

Sehr bald erhielt der CERN-Webserver allerdings Konkurrenz aus den USA: Am National Center for Supercomputing Applications (NCSA) der University of Illinois wurde neben Mosaic – dem Urahn aller modernen GUI-basierten Webbrowser – auch ein erweiterter HTTP-Server programmiert, eben der NCSA HTTPd. Als der Leiter dieses Projekts, Rob McCool, das NCSA 1994 verließ, wurde dieser Webserver nicht mehr weiter gepflegt. Aber einige der Programmierer, die daran mitgearbeitet hatten, blieben per E-Mail in Kontakt. Sie bildeten die ursprüngliche Apache Group:

- Brian Behlendorf
- Roy T. Fielding
- Rob Hartill
- David Robinson
- Cliff Skolnick
- Randy Terbush
- Robert S. Thau
- Andrew Wilson

Weitere Beiträge zur ursprünglichen Version von Apache haben folgende Entwickler geleistet:

- Eric Hagberg
- Frank Peters
- Nicolas Pioch

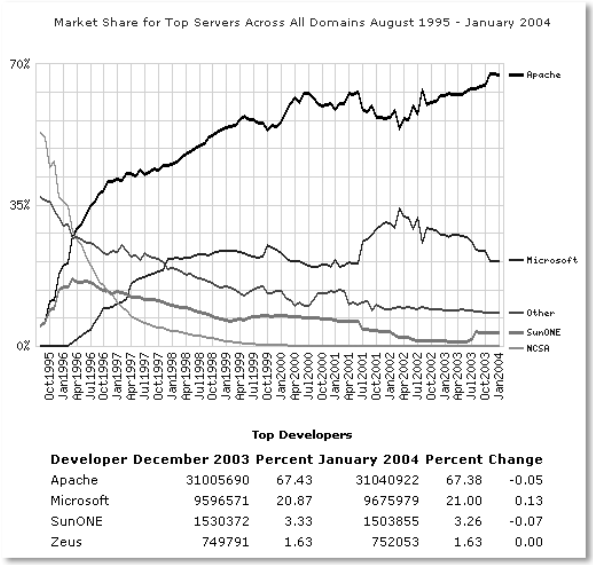
Im Laufe der Jahre haben hunderte von Personen an der Weiterentwicklung des Servers mitgearbeitet. Eine aktuelle Liste der Mitglieder der Apache Group finden Sie im Web unter **httpd.apache.org/contributors/**.

Auf diese Weise entwickelte sich aus der Version 1.3 des NCSA HTTPd und zahlreichen von diesen Entwicklern geschriebenen Patches (»Flicken«) »a PATCHy web server« (ein Webserver mit Patches). Laut Apache-FAQ (**httpd.apache.org/docs/misc/FAQ.html**) ist dies allerdings nicht der eigentliche Grund für die Namenswahl (auch wenn er dort verdächtigerweise erwähnt wird):

»The name ›Apache‹ was chosen from respect for the Native American Indian tribe of Apache (Indé), well-known for their superior skills in warfare strategy and their inexhaustible endurance.«[3]

Die erste offizielle Release des Apache-Webservers (1.0) wurde im Dezember 1995 veröffentlicht. Die ursprüngliche Codebasis (1.x) des Webservers wurde bis zur Version 1.3 weitergeführt. Inzwischen wird sie im Wesentlichen nur noch durch Bugfixes ergänzt. Die aktuelle Unterversion ist 1.3.29. Apache 1.3.x ist aber nicht Thema dieses Buches; hier geht es um die von Grund auf weiterentwickelte Version 2.0. Dennoch gelten viele Themen auch für 1.3; zusätzlich habe ich in Anhang A die wichtigsten Besonderheiten der alten Version zusammengefasst, weil in manchen Produktionsumgebungen noch damit gearbeitet wird.

Bereits ein Jahr nach seiner Einführung überholte Apache den NCSA HTTPd als meistgenutzter Server im Internet. Dies ist bis heute unverändert geblieben: Im Januar 2004 dominierte der Einsatz des Apache-Webservers mit einem Anteil von 67 Prozent aller weltweit erreichbaren Websites weit vor dem Microsoft Internet Information Server (IIS), der mit 21 Prozent einen ebenso einsamen zweiten Platz einnimmt. Abbildung 3.1 zeigt die Entwicklung der Webserver-Anteile, seit Netcraft (**www.netcraft.com**) im Jahr 1995 mit deren Erhebung begonnen hat.




**Top Developers**

| Developer | December 2003 | Percent | January 2004 | Percent | Change |
|---|---|---|---|---|---|
| Apache | 31005690 | 67.43 | 31040922 | 67.38 | -0.05 |
| Microsoft | 9596571 | 20.87 | 9675979 | 21.00 | 0.13 |
| SunONE | 1530372 | 3.33 | 1503855 | 3.26 | -0.07 |
| Zeus | 749791 | 1.63 | 752053 | 1.63 | 0.00 |

**Abbildung 3.1** Entwicklung der Webserver-Marktanteile seit 1995. Quelle: Netcraft

3 Übersetzung: »Der Name ›Apache‹ wurde aus Respekt vor dem amerikanischen Indianerstamm der Apachen (Indé) gewählt, die für ihre außergewöhnlichen Fähigkeiten in der Strategie der Kriegsführung und ihre unerschöpfliche Ausdauer berühmt waren.«

### 3.1.2 Die Apache Software Foundation

Um die Entwicklung des Apache-Webservers auf eine solide Basis zu stellen, wurde 1999 die Apache Software Foundation (ASF) als »Dachorganisation« der Apache Group gegründet. Sie kümmert sich seitdem um zahlreiche Projekte freier Software; die wichtigsten von ihnen werden weiter unten kurz vorgestellt. Eine weitere Aufgabe der ASF besteht darin, die Markenrechte an dem Namen »Apache« und den sonstigen Projekten zu wahren.

Die Foundation definiert sich selbst als »Meritocracy« – Mitglied wird, wer reichlich an Projekten mitgearbeitet hat und dann von einem bestehenden Mitglied vorgeschlagen und von einer Mehrheit gewählt wird.

Unter dem Dach der ASF werden neben dem HTTP-Server zurzeit folgende Projekte gepflegt:

- **Ant.** Ein Compiler-Fernsteuerungs- und Software-Projektmanagement-Tool für Java auf XML-Basis, also ein moderner Java-Ersatz für `make`. Die Website des Projekts finden Sie unter **ant.apache.org**.
- **APR.** Die **Apache Portable Runtime** wird weiter unten noch genauer erläutert – diese API liegt dem Apache-Webserver zugrunde und abstrahiert einige System- und I/O-Funktionen. Website: **apr.apache.org**.
- **Avalon.** Ein modernes Komponenten-Framework für Java zur effizienten Softwareentwicklung. Es basiert darauf, die Funktionalität von Software in einzelne Rollen (Roles) und Dienste (Services) zu zerlegen und als kooperative Einzelkomponenten zu erstellen. Die zugehörige Website ist **avalon.apache.org**.
- **Cocoon.** Framework für XML- und XSLT-basierte Server-Anwendungen, das unter anderem mit Tomcat zusammenarbeitet. Alles über das Projekt erfahren Sie unter **cocoon.apache.org**.
- **Commons.** Ein neues Projekt in der Apache Software Foundation, das es sich zur Aufgabe gemacht hat, Bibliotheken und Komponenten für verschiedene Programmiersprachen zu schaffen, die in unterschiedlichen Softwareprojekten (zum Beispiel von der ASF selbst) wiederverwendet werden können. Die Site ist natürlich **commons.apache.org**.
- **DB.** Dieses Projekt kümmert sich um verschiedene Lösungen zur Datenbankprogrammierung und -integration. Bemerkenswert ist zum Beispiel die **ObjectRelationalBridge**, ein Programm zur automatischen, persistenten Abbildung relationaler Datenbanken in Java-Objekten. Downloads und weitere Informationen gibt es unter **db.apache.org**.

- **Incubator.** Der »Brutkasten« ist kein Softwareprojekt, sondern eine Gruppe innerhalb der Apache Software Foundation, die sich um die Aufnahme neuer Projekte unter das Dach der Foundation kümmert. Vielleicht möchten Sie sich ja eines Tages mit einem viel versprechenden Open-Source-Projekt selbst an die Apache Software Foundation wenden? Dann erfahren Sie alles Nötige unter **incubator.apache.org**.
- **Jakarta.** Jakarta ist ein gemeinsames Dach für zahlreiche Java-Softwareprojekte in der ASF. Einige Beispiele sind der JSP- und Servlet-Server **Tomcat** (siehe Kapitel 13, *Technologien zur Webprogrammierung*), das Webanwendungs-Framework **Struts** oder die Dokumentationsverwaltung **Alexandria**. Die Jakarta-Website mit Links auf die zahlreichen Unterprojekte finden Sie unter **jakarta.apache.org**.
- **James.** Der **Java Apache Mail Enterprise Server** ist ein vollständig in Java geschriebener SMTP-, POP3- und NNTP-News-Server. Das Projekt finden Sie unter **james.apache.org**.
- **Logging.** Da Logdateien nicht nur Systemadministratoren, sondern auch Programmierern wertvolle Informationen zur Fehlersuche liefern können, ist es wünschenswert, eine einheitliche Logging-Schnittstelle zur Verfügung zu haben. Das Projekt `log4j` bietet eine solche für Java. Seit es als Logging-Projekt in die ASF aufgenommen wurde, begann man dort mit der Entwicklung ähnlicher Hilfsmittel für C, C++, .NET, PHP und Perl. Website: **logging.apache.org**.
- **Maven.** Dies ist ein Java-Projektmanagement-Paket. Es enthält ein CVS-ähnliches Repository, Ant-artige Build-Werkzeuge und weitere Features und basiert auf einem Project Object Model (POM), das bisher im XML-Format gespeichert wird, zukünftig aber beispielsweise auch in einer relationalen Datenbank abgelegt werden soll. Näheres erfahren Sie unter **maven.apache.org**.
- **Perl.** Hinter dem schlichten Projektnamen »Perl« verbirgt sich das mächtige Webserver-Modul `mod_perl`. Es zählt zu den beliebtesten Apache-Modulen überhaupt, weil es nicht nur das Schreiben von Webanwendungen in Perl, sondern auch die Erweiterung des Servers selbst durch Perl-Module ermöglicht. Natürlich hat das Modul in diesem Buch sein eigenes Kapitel: Das Wichtigste über `mod_perl` lesen Sie in Kapitel 13. Noch genauere Informationen gibt es auf der Site **perl.apache.org**.
- **PHP.** Die Programmiersprache PHP (ein rekursives Akronym für PHP: Hypertext Preprocessor) bietet eine praktische Schnittstelle für die Entwicklung von Webanwendungen. Zahlreiche mächtige Funktionen und vor allem der einfache Zugriff auf zahlreiche Datenbanksysteme machen PHP zu einer der

beliebtesten Plattformen für die serverseitige Webprogrammierung. Die Integration in den Apache-Webserver erfolgt entweder über die klassische CGI-Schnittstelle oder mit Hilfe des Moduls `mod_cgi`. Beides wird in Kapitel 13 genau erläutert. Die Website des Projekts ist **www.php.net**.

- **TCL.** Diese klassische Skriptsprache besitzt eine recht einfache Syntax. Das Apache-Projekt bietet Schnittstellen zur Integration der Sprache für Webserver-Anwendungen über CGI und das Modul `mod_tcl`. Die Projektwebsite ist **tcl.apache.org**.
- **Web Services.** Mit Hilfe von Web Services können Anwendungen unter Verwendung von HTTP über das Web miteinander kommunizieren. Diese Anwendungen können in den unterschiedlichsten Sprachen programmiert werden – Web-Service-APIs gibt es unter anderem für Java, PHP, Perl oder das Microsoft .NET Framework. Das Apache-Web-Services-Projekt bietet freie Implementierungen dieser APIs für verschiedene Programmiersprachen. Website: **ws.apache.org**
- **XML.** Die **eXtensible Markup Language** hat es seit ihrer Einführung 1998 zum weithin anerkannten Standard für die Strukturierung von Daten aller Art gebracht. Das Apache-XML-Projekt bietet Parser, XSLT-Prozessoren und XML-APIs für verschiedene Programmiersprachen. Das Projekt befindet sich unter **xml.apache.org**.

### 3.1.3 Die Apache-Softwarelizenz

Der Begriff »freie Software« steht nicht nur dafür, dass der Quellcode frei verfügbar ist, sondern er bezeichnet allgemein Computerprogramme, mit denen Anwender »alles« machen dürfen. Dies betont den Gegensatz zu kommerzieller Software, die meist unter einer sehr restriktiven Lizenz steht: Ein von Juristen formulierter Lizenzvertrag (meist »EULA« oder End User Licence Agreement genannt) regelt detailliert, was Sie mit einem solchen Programm anstellen dürfen. Neben dem Kopieren über eine persönliche Sicherheitskopie hinaus ist es meist vor allem untersagt, die Software selbst näher zu untersuchen oder gar durch Reverse Engineering ihre genaue Funktionsweise nachzuahmen.

Solchen Einschränkungen ist freie Software nicht unterworfen. Das bedeutet aber keineswegs, dass sie vollkommen rechtsfrei veröffentlicht wird. Zunächst einmal gilt natürlich das Urheberrecht der eigentlichen Autoren, wobei die meisten freien Softwareprojekte von zahlreichen über die ganze Welt verteilten Entwicklern geschrieben werden. Abgesehen davon stehen auch die meisten Open-Source-Projekte unter Softwarelizenzen. Diese werden natürlich ebenfalls von Juristen formuliert, dienen aber einem anderen Zweck: Es muss sichergestellt werden, dass sich kein kommerzieller Softwarehersteller das Pro-

gramm exklusiv zu eigen macht oder gar zum Patent anmeldet. Es gibt unterschiedliche Lizenzen für freie Software; die Lizenz der Apache Software Foundation ist nur eine von ihnen. Andere bekannte Open-Source-Lizenzen sind folgende:

- Die **GNU General Public License** (GPL) der Free Software Foundation ist die älteste bekannte Lizenz für freie Software. Sie gilt für sämtliche Software des GNU-Projekts und beispielsweise auch für den Linux-Kernel. Natürlich erlaubt sie die beliebige Weitergabe und Modifikation der Software, untersagt aber grundsätzlich deren Verwendung in kommerziellen Produkten. Da dies im Falle von Compilern oder Bibliotheken zu Problemen führen kann, existiert daneben die LGPL (Library oder Lesser GPL), die diesen Einsatz in engen Grenzen gestattet.
- Die **BSD-Lizenz**, unter der vor allem die freien BSD-UNIX-Varianten FreeBSD, OpenBSD und NetBSD verbreitet werden, kann als Gegenteil der restriktiven GPL betrachtet werden: Sie erlaubt den Einsatz der Software unter fast beliebigen Bedingungen.

Die Apache-Softwarelizenz liegt etwa in der Mitte zwischen diesen beiden Extremen.[4] Sie gestattet die Verwendung, Weitergabe und Veränderung von Software der ASF unter einigen Bedingungen. Da die Lizenzvereinbarung im Gegensatz zur mehrere Seiten füllenden GPL überraschend kurz ist, ist es kein Problem, sie hier in ihrer vollen Länge wiederzugeben. Damit sie in den Satzspiegel dieses Buches passt, habe ich mir erlaubt, die Zeilenumbrüche neu zu setzen.

```
/* ====================================================================
 * The Apache Software License, Version 1.1
 *
 * Copyright (c) 2000-2004 The Apache Software Foundation.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or
 * without modification, are permitted provided that the
 * following conditions are met:
 *
 * 1. Redistributions of source code must retain the above
 *    copyright notice, this list of conditions and the following
 *    disclaimer.
 *
```

4 Genauer gesagt ähnelt die hier gezeigte alte Fassung eher der BSD-Lizenz, während die neue Version (siehe Anhang D) sich stärker der GPL angenähert hat.

```
* 2. Redistributions in binary form must reproduce the above
*    copyright notice, this list of conditions and the following
*    disclaimer in the documentation and/or other materials
*    provided with the distribution.
*
* 3. The end-user documentation included with the
*    redistribution, if any, must include the following
*    acknowledgment:
*       "This product includes software developed by the
*        Apache Software Foundation (http://www.apache.org/)."
*    Alternately, this acknowledgment may appear in the software
*    itself, if and wherever such third-party acknowledgments
*    normally appear.
*
* 4. The names "Apache" and "Apache Software Foundation" must
*    not be used to endorse or promote products derived from
*    this software without prior written permission. For written
*    permission, please contact apache@apache.org.
*
* 5. Products derived from this software may not be called
*    "Apache", nor may "Apache" appear in their name, without
*    prior written permission of the Apache Software Foundation.
*
* THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR
* IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE
* FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
* OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
* ====================================================================
*
* This software consists of voluntary contributions made by many
* individuals on behalf of the Apache Software Foundation.  For
* more information on the Apache Software Foundation, please see
* <http://www.apache.org/>.
*
```

```
 * Portions of this software are based upon public domain
 * software originally written at the National Center for
 * Supercomputing Applications, University of Illinois,
 * Urbana-Champaign.
 */
```

Die fünf nummerierten Punkte der Lizenzvereinbarung bedeuten kurz gesagt Folgendes:

1. Bei einer Weitergabe des Quellcodes der Software muss die vollständige Lizenz einschließlich Copyright und Haftungsausschluss mitgeliefert werden.
2. Auch bei der Weitergabe in binärer Form muss die Lizenzvereinbarung enthalten sein, beispielsweise in der mitgelieferten Dokumentation.
3. Bei Weiterverwendung in eigenen Projekten muss die Dokumentation oder die Bildschirmausgabe den Hinweis enthalten, dass das Programm Software der Apache Software Foundation enthält.
4. Ohne Erlaubnis der ASF darf der Name »Apache« oder »Apache Software Foundation« nicht verwendet werden, um für eigene, davon abgeleitete Produkte zu werben.
5. Außerdem dürfen Projekte, die aus der Weiterentwicklung eines ASF-Programms entstanden sind, nicht Apache heißen oder das Wort »Apache« im Namen enthalten.

Der Rest des Textes ist ein Haftungsausschluss (Disclaimer), der verhindert, dass die Entwickler der Software oder die Apache Software Foundation Schadenersatz leisten müssen, wenn das Programm nicht richtig funktionieren sollte oder im Zusammenhang mit der Verwendung des Programms Datenverluste oder andere Schäden auftreten sollten.

Im Januar 2004 wurde die neue Version 2.0 der Apache-Lizenz eingeführt. Sie ist länger als die hier zitierte Version 1.1, weil sie viele Formulierungen präzisiert und einige Punkte ergänzt, etwa um den schädlichen Einfluss der ärgerlichen Softwarepatente zu minimieren. Wegen ihres Umfangs habe ich die neue Lizenz in Anhang D verschoben.

### 3.1.4 Sonstige Webserver

Neben Apache gibt es natürlich noch einige andere Webserver – sowohl freie als auch kommerzielle. Einige von ihnen sollen an dieser Stelle kurz vorgestellt werden, um einen Vergleich mit dem Apache-Webserver zu bieten.

#### Microsoft Internet Information Server

Dies ist der Webserver von Microsoft. Er wird nicht als eigenständiges Programmpaket angeboten, sondern ist seit Windows NT Server 4.0 aus dem Jahr 1996 Teil der Microsoft-Server-Betriebssysteme. Er läuft also nur unter Windows und nicht unter einer Vielzahl von Plattformen wie Apache. Das aktuellste Server-Betriebssystem, Windows Server 2003, enthält den IIS in der neuesten Version 6.0 und wird unter anderem in Form einer speziellen **Web Edition** verkauft, die auf viele andere Server-Komponenten wie Active-Directory-Domänen-Controller, DNS-Server oder Ähnliches verzichtet und sich statt dessen auf die Aufgabe als Webserver und .NET-Application-Server konzentriert. Dafür ist diese Ausgabe erheblich billiger als die Varianten Standard oder gar Enterprise.

Der Internet Information Server ist übrigens nicht nur ein Webserver, sondern enthält auch einen eingebauten FTP-, einen News- und einen SMTP-Server. Alle diese Komponenten lassen sich über ein grafisches Tool konfigurieren; eine Konfigurationsdatei nach UNIX- oder Apache-Art gibt es nicht. Dies erleichtert zwar den Überblick über die verfügbaren Konfigurationsoptionen, aber es macht die Automatisierung und Übertragbarkeit der Konfigurationsdateien kompliziert.

Natürlich gibt es einige spezielle Vorteile, die nur dieser Server aufgrund seiner Integration in die Windows-Funktionalität zu bieten hat. Dazu gehören unter anderem folgende Punkte:

- **Active Server Pages.** Dies ist Microsofts Schnittstelle für Skriptbefehle, die unmittelbar in HTML-Dokumente hineingeschrieben und vom Server ausgeführt werden, bevor das Dokument an den Client geliefert wird. Das Konzept ähnelt somit PHP oder Java ServerPages (JSP). Die ursprünglichen ASP wurden in der Sprache VBScript geschrieben, einer einfachen Skriptvariante von Microsoft Visual Basic. Die aktuelle Variante ASP.NET lässt sich dagegen in allen Sprachen programmieren, die die Common Language Runtime (CLR) des .NET Frameworks verwenden kann. Dies sind von Microsofts Seite Visual Basic .NET, Visual C++ und C#.
- **ISAPI.** Der IIS ist mit einer speziellen Schnittstelle namens Internet Server API (ISAPI) ausgestattet. Diese ermöglicht das Schreiben von Erweiterungen, die als DLL-Dateien kompiliert werden und somit schneller laufen als ASP

oder CGI. Für die Windows-Version von Apache steht übrigens ebenfalls eine ISAPI-Implementierung in Form des Moduls `mod_isapi` zur Verfügung. Die Performance dieses Moduls ist allerdings nicht mit dem Original-ISAPI vergleichbar. Außerdem ermöglicht die Apache-Variante keine ISAPI-Filter.

- **Windows- und .NET-Integration.** Da es sich beim IIS um eine Technologie von Microsoft handelt, lässt er sich besser in die Windows-Systemumgebung integrieren als Apache. Sie haben mit Hilfe der erwähnten ASP.NET-Schnittstellen und ISAPI Zugriff auf das gesamte Leistungsspektrum eines Windows-Server-Systems und auf das .NET Framework: Beispielsweise können Sie über ADO.NET auf Datenbanken und XML zugreifen oder Authentifizierungen über Active Directory durchführen.

Fazit: In einer bestehenden Microsoft-Server-Umgebung bietet sich die Anwendung des Internet Information Servers einfach an. In diesem speziellen Fall ist selbst das Kostenargument hinfällig: Wenn Sie bereits Server-Betriebssysteme von Microsoft betreiben, ist der Internet Information Server automatisch darin enthalten und verursacht keine Zusatzkosten. Der Nutzen, den die Integration in die vorhandene Infrastruktur bietet, übersteigt hier den möglichen Sicherheits- und Performancegewinn durch Apache.

Die Konfiguration des IIS erfolgt über die grafische Benutzeroberfläche. Abbildung 3.2 zeigt die Haupt-Konfigurationsseite für den Server, die Sie über **Start · Verwaltung · Internetinformationsdienste-Manager** erreichen.

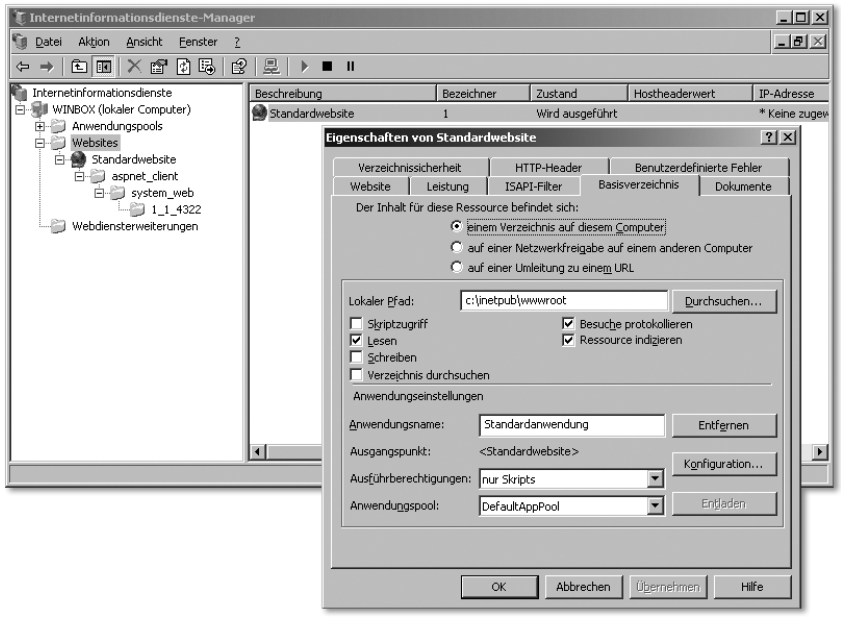



**Abbildung 3.2** Der Haupt-Konfigurationsdialog des Microsoft Internet Information Servers

### Zeus

Der Zeus Web Server von Zeus Technologies, Inc. ist ein kommerzieller Webserver für UNIX-Systeme, der besonders für möglichst hohe Belastung und Performance optimiert wurde. Eine der bekanntesten Websites, die auf Zeus setzen, ist Ebay.

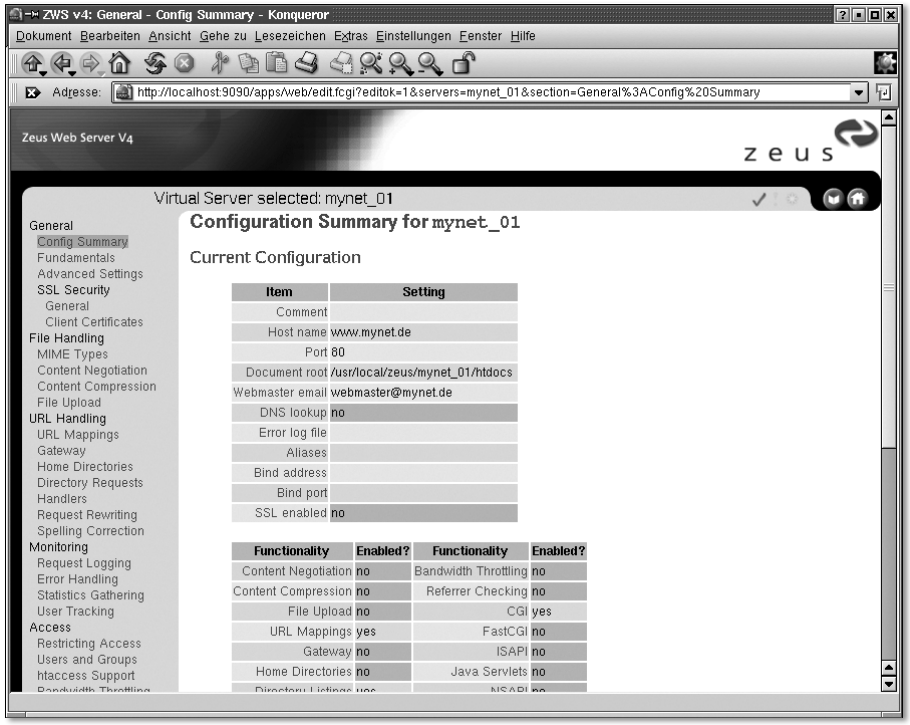



**Abbildung 3.3** Die webbasierte Konfiguration des HTTP-Servers Zeus

Der Zeus-Webserver wird über ein einfaches Shell-Skript installiert. Es stellt Ihnen einige Konfigurationsfragen, etwa nach Installationspfad, Administratorpasswort und so weiter. Nachdem der Server installiert ist, können Sie seine webbasierte Konfiguration aufrufen. Öffnen Sie dazu einen beliebigen Browser und geben Sie als URL **http://localhost:9090** (beziehungsweise Port 9090 auf dem entfernten Host, auf dem Zeus ausgeführt wird) ein. Nun müssen Sie sich mit dem Benutzernamen **admin** und dem bei der Installation festgelegten Passwort anmelden. Die eigentlichen Websites werden bei Zeus stets als virtuelle Hosts eingerichtet. Abbildung 3.3 zeigt die Konfigurationsseite für einen solchen Server. Bequemerweise lässt sich ein vorhandener virtueller Host duplizieren, wenn Sie einen neuen einrichten möchten, der nur geringfügig vom vorherigen abweichen soll.

### Tomcat

Jakarta Tomcat, wie bereits erwähnt ebenfalls ein Projekt der Apache Software Foundation, wird in Kapitel 13, *Technologien zur Webprogrammierung*, genauer behandelt. Während dort vor allem auf seine Fähigkeiten als Java-Servlet- und JSP-Engine eingegangen wird, die mit einem Apache-HTTP-Server zusammenarbeitet, soll nicht unerwähnt bleiben, dass Tomcat selbst ebenfalls ein recht leistungsfähiger Webserver ist. Natürlich kommt er in Performance und Leistungsspektrum nicht an den Apache-Webserver heran, aber das ist auch gar nicht die Intention seiner Entwickler. Für Websites, die nur in geringem Umfang aus statischen HTML-Dokumenten bestehen und fast ausschließlich auf Java-Technologie setzen, dürfte eine Tomcat-Standalone-Lösung ideal sein. Für alle anderen Sites, die nur teilweise Servlets oder JSP einsetzen, existieren zahlreiche Möglichkeiten, Tomcat in Kombination mit Apache 2 zu betreiben.

Auch wenn der Schwerpunkt von Kapitel 13 aufgrund der Thematik dieses Buches ein anderer ist, erhalten Sie dort einen kurzen Überblick über die Verwendung von Tomcat als Webserver. Anders als der Apache-HTTP-Server, der aus Kompatibilitätsgründen das historisch gewachsene Format der NCSA-Konfigurationsdateien übernommen hat, verwaltet Tomcat seine Einstellungen konsequent in XML-Dateien.

## 3.2 Funktionen von Apache 2

Die Informationen über die Installation, Konfiguration und Programmierung des Apache-HTTP-Servers in den restlichen Kapiteln dieses Buches betreffen eher einzelne Aspekte. Deshalb erhalten Sie an dieser Stelle zunächst einen Gesamteindruck über seine Funktionsweise und seinen Leistungsumfang.

Es ist keine große Überraschung, dass eine vollständige und vorbildliche Implementierung des HTTP/1.1-Standards gemäß RFC 2616, wie sie im vorigen Kapitel beschrieben wurde, den Kern von Apache 2 bildet. Daneben gibt es unzählige Erweiterungen für Spezialaufgaben wie die Zusammenarbeit mit bestimmten Skriptsprachen, URL-Umleitungen, erweitertes Logging oder sogar die Unterstützung anderer Protokolle außer HTTP (Apache als Proxy-Server, FTP-Server oder gar Mailserver). Alle diese Erweiterungen sind in Form von Modulen realisiert, die sich entweder statisch einkompilieren oder dynamisch zur Laufzeit hinzuladen lassen.

Die ursprüngliche Fassung des Servers wurde ausschließlich für UNIX-Systeme entwickelt. Erst ab Version 1.2 wurde mit der Portierung auf andere Betriebssysteme begonnen. Durch die Schaffung der weiter unten näher behandelten Apache Portable Runtime (APR), einer virtuellen Systemschicht, konnte die Stabilität und Performance der Portierungen in Version 2 entscheidend verbessert werden.

Apache 2 wird (mindestens) von folgenden Systemen der großen UNIX-Familie unterstützt:

- Linux
- FreeBSD
- OpenBSD
- NetBSD
- Sun Solaris
- IBM AIX
- HP UX
- BeOS
- Mac OS X

Auf den meisten anderen modernen UNIX-Systemen müsste sich Apache 2.0 ebenfalls problemlos kompilieren lassen.

Daneben wurde Apache erfolgreich auf folgende Nicht-UNIX-Plattformen portiert:

- Microsoft-Windows-NT-Familie (NT 4.0, 2000, XP)
- Microsoft-Windows-9x-Familie (95, 98, Me)
- Novell NetWare
- VMS und OpenVMS
- IBM OS/2
- BS2000

Diese Plattformliste ist mit Sicherheit nicht vollständig – zumindest Apache 1.3 wurde beispielsweise auch auf AmigaOS portiert. Apache ist freie Software, und im Grunde kann jeder Interessierte versuchen, das Projekt auf eine neue Plattform zu portieren. Unterstützt wird dies dadurch, dass auch die GNU-Entwicklungswerkzeuge für immer mehr Nicht-UNIX-Plattformen zur Verfügung stehen.

**Welche Plattform ist zu empfehlen?**

Auf diese Frage sollte man am besten die vorsichtige Antwort des Juristen geben: »Es kommt darauf an.« Bei weitem die meisten Apache-Webserver im Internet sind auf UNIX-Systemen installiert, allen voran Linux und die freien BSD-Varianten.

Das Internet wurde auf UNIX-Maschinen zu dem, was es heute ist, und Apache (mitsamt seinem Vorläufer NCSA HTTPd) ist auf dieser Plattform entstanden. Unterstützung für andere Systeme war am Anfang experimentell – selbst frühe Releases von Apache 1.3 für Windows wurden noch mit dem ausdrücklichen Warnhinweis versehen, dass die Windows-Version als eine Art Betaversion zu verstehen sei und man von ihr bei weitem nicht den von UNIX gewohnten Standard erwarten dürfe.

Mit der Veröffentlichung von Apache 2.0 hat sich dies allerdings grundlegend geändert: Durch die Apache Portable Runtime (siehe unten) und die unterschiedlichen Multiprocessing-Module konnte die Performance und Stabilität des Servers auch auf Nicht-UNIX-Systemen erheblich gesteigert werden. Deshalb kann die Empfehlung heutzutage im Grunde lauten: Verwenden Sie Apache einfach auf der Plattform, die zu Ihrer IT-Infrastruktur passt. Wenn Sie ein Windows-basiertes Netzwerk betreiben, können Sie den Server genauso gut auf einem Windows-2000- oder XP-Rechner betreiben wie auf einem UNIX-System. Sollten Ihre Server unter Novell NetWare arbeiten, verwenden Sie doch einfach die passende Apache-Version.

Gemäß der praktischen Verbreitung des Webservers konzentriert sich dieses Buch auf die Apache-Versionen für UNIX-Systeme – die man in diesem Fall trotz gewisser Unterschiede über einen Kamm scheren kann – und die Windows-NT-Familie.

**Vorsicht!**

Sie sollten aus Performance- und vor allem Sicherheitsgründen **dringend** davon absehen, einen Produktions-Server unter Windows 9x zu betreiben! Natürlich hält Sie niemand davon ab, in einem kleinen bis mittleren Unternehmen einen alten Pentium-I-Rechner in die Ecke zu stellen, Windows 98 und Apache zu installieren und damit im Intranet E-Books, Dokumentationen oder MP3-Dateien für die Mittagspausenunterhaltung zu verbreiten. Aber selbst eine solche altersschwache Maschine läuft prima unter Linux, und es schadet überhaupt nichts, das entsprechende Plus an Sicherheit mitzunehmen – eine Linux-auf-einer-CD-Distribution wie **Knoppix** (**www.knopper.net/knoppix**) reicht dazu vollkommen aus.

### 3.2.1 Technischer Überblick

Behandelt wird in diesem Buch die aktuelle, stabile Version des Apache-HTTP-Servers. Das ist heute (März 2004) die Version 2.0, genauer 2.0.49. Einige

Informationen über den noch immer weit verbreiteten Apache 1.3 finden Sie in Anhang A.

Hier einige der überzeugendsten Gründe dafür, warum Sie Apache als Webserver verwenden sollten:

- **Verbreitung.** Wie bereits erwähnt, ist Apache der mit Abstand am häufigsten eingesetzte Webserver der Welt. Das heißt, dass kein anderer HTTP-Server so vielen Praxistests unterworfen wurde. Apache ist somit nachgewiesenermaßen ein praxistauglicher Allzweck-Webserver. Dies gilt inzwischen auch für Apache 2, der auf immer mehr Produktions-Servern die frühere Version 1.3 ersetzt. Darüber hinaus gewährleistet der hohe Verbreitungsgrad den Zugriff auf Unmengen an Dokumentation und öffentlichem Support.
- **Performance.** Es mag vereinzelt unsäglich teure kommerzielle Webserver geben, die ein wenig schneller sind als Apache. Durch wirklich neutrale und reproduzierbare Benchmarks wurde dies allerdings noch nicht nachgewiesen; wie bei anderen Server-Diensten, Betriebssystemen oder Hardware pflegen sich die kommerziellen Hersteller stark angepasste Umstände zu schaffen, unter denen ihr Produkt schneller läuft. Die Apache Group dagegen hat für so etwas keine Zeit und arbeitet lieber an der Leistungsfähigkeit des eigenen Produkts.[5] Auch hier gilt der Verbreitungsgrad als Nachweis: Es gibt zahllose große Websites mit Millionen von Zugriffen am Tag, die Apache einsetzen und deren gute Performance für 99,9 Prozent aller Anfragen gewährleistet ist.

  Die Performance von Apache 2.0 wurde gegenüber früheren Versionen noch einmal erheblich gesteigert – und zwar auch für Nicht-UNIX-Plattformen. Das liegt vor allem an der Einführung der Apache Portable Runtime (APR) als solider Plattform-Abstraktions-API sowie an den verschiedenen Multiprocessing-Modulen, die die Leistung für unterschiedliche Einsatzzwecke verbessern.
- **Stabilität.** Bei der Weiterentwicklung des Apache-Webservers wird vor allem auf stabile und robuste Funktionsfähigkeit Wert gelegt. Besonders der modulare Aufbau unterstützt dies: Ein schlankes, leistungsfähiges Grundprogramm mit wenigen Funktionen wird durch zahllose, mit Hilfe einer sauberen API angebundene Module ergänzt.

5 Nichtsdestotrotz gibt es ein Unterprojekt der Apache Group, das sich mit der Bereitstellung von Benchmark-Tools beschäftigt. Diese versuchen aber eben gerade nicht, sich selbst (und die potentiellen Anwender) zu betrügen, sondern möglichst neutrale und aussagekräftige Daten zu liefern. Sie dienen auch eher dem Performance-Tuning einer bestehenden Apache-Installation als dem Wettbewerb.

- **Sicherheit.** Apache ist freie Software. Anders als bei kommerziellen Programmen mit einem in sich geschlossenen Entwicklungsteam wird der Quellcode hier von zahlreichen über die ganze Welt verteilten Programmierern mit den unterschiedlichsten Erfahrungen und Umfeldern geschrieben und begutachtet. So werden Fehler in aller Regel sehr schnell gefunden und beseitigt. Dank des verfügbaren Quellcodes kann zur Not sogar jeder mit entsprechenden C-Kenntnissen selbst einen Bug beseitigen, wenn es schnell gehen muss. Da viele Apache-Anwender dies tun und ihre Bugfixes anschließend oft per Mailing-Liste an die Apache Group weitergeben, steigt die Chance für schnelle Fehlerbereinigungen sogar noch an. Da Fehler im Programmcode (neben schlechter Konfiguration, die dieses Buch hoffentlich verbessern hilft) das größte Sicherheitsrisiko darstellen, kann man ein Open-Source-Projekt wie den Apache-Webserver als sehr sicher einstufen.[6]
- **Skalierbarkeit.** Mit Apache können Sie Sites in völlig beliebiger Größe und Komplexität betreiben: Angefangen von einem kleinen Intranet-Webserver mit wenigen hundert Dokumenten über eine mittlere Firmenwebsite bis hin zu einem riesigen Server-Raum voller Load-Balancing-Server-Rechner oder virtueller Hosts. Für den erstgenannten Zweck kann Apache 2 fast ohne erkennbaren Ressourcenverbrauch auf einem Desktop-PC im Hintergrund laufen. Sie können den Server an eine wachsende Website anpassen, indem Sie die Konfigurationsdatei, die Sie im Laufe der Zeit Ihren Bedürfnissen angepasst haben, auf eine größere Maschine mitnehmen – der wesentliche Aufbau der Datei ist über alle System- und Plattformunterschiede hinweg derselbe.

### Apache-Versionierung

Ab der neuen Version 2.0 hat auch die Apache Group die Versionszählweise anderer freier Softwareprojekte wie Linux (und Perl seit Version 5.6) übernommen: Eine gerade Unterversionsnummer (2.0, 2.2 und so weiter) steht für die stabilen Releases, während ungerade Nummern, also 2.1 oder 2.3, die jeweilige Entwicklerversion bezeichnet: Zurzeit ist 2.0 die stabile Version; an der Entwicklerversion 2.1 wird gerade gearbeitet – wenn sie irgendwann fertig ist, wird sie als 2.2 veröffentlicht, und die Entwickler werden mit der Arbeit an 2.3 beginnen. Ob diese dann (schätzungsweise erst in einigen Jahren) als Version 2.4 oder als 3.0 veröffentlicht wird, weiß heute natürlich noch niemand – neue

6 Wenn Sie Sicherheits-Mailing-Listen wie BugTraq lesen (was Sie als verantwortungsbewusster Administrator natürlich tun sollten), werden Sie fast täglich Meldungen über neue Apache-Sicherheitslücken erhalten. Die Sicherheit wird nämlich nicht durch Software ohne Fehler gewährleistet (die gibt es nicht!), sondern durch ein effizientes und vernetztes Finden und Beseitigen von Fehlern.

Hauptversionsnummern werden bei freier Software eher sparsam vergeben, nämlich im Grunde nur dann, wenn erhebliche Neuentwicklungen durchgeführt wurden.

Manchmal kommt es dagegen vor, dass neue Errungenschaften der Entwicklerversion nachträglich in neuere Releases der vorigen stabilen Version übernommen werden (so genannte Backports) – dies geht natürlich nur dann, wenn sie grundsätzlich zur alten Version kompatibel sind.

Dieser sparsame und vorsichtige Umgang mit Versionsnummern und Veröffentlichungen führt natürlich zu größerer Anwenderzufriedenheit: Sie können fast immer ein einigermaßen ausgereiftes, stabiles und fehlerarmes Programm erwarten, wenn Sie sich eine stabile Release besorgen. Bei kommerziellen Softwareherstellern sieht das leider oft anders aus: Manche bringen innerhalb weniger Monate schon wieder die nächste Hauptversion heraus, die sich dann als »Banana Ware« entpuppt: Sie ist im Auslieferungszustand noch »grün« und »reift beim Kunden«.

Betrachten Sie zum Vergleich die Versionsnummern des Linux-Kernels: Allein der Sprung von Version 2.4 (Ende 2001) bis zum brandaktuellen Kernel 2.6 hat ganze drei Jahre gedauert.

Aber selbst wenn eine neue stabile Version veröffentlicht wird, steigt nicht unbedingt alle Welt von heute auf morgen darauf um. Das hat mitunter wichtige praktische Gründe: Bei Apache ist beispielsweise die Modularchitektur der Vorgängerversion 1.3 inkompatibel zum neuen 2.0-Format. Zwar wurde 2.0 erst als stabil gekennzeichnet, als die wichtigsten Module der Apache Group selbst portiert waren, aber Drittanbietermodule wurden erst allmählich (oder gar nicht) auf die neue Version zugeschnitten. Wer also eines oder mehrere wichtige Module im Einsatz hatte, die nicht für Apache 2 verfügbar waren, musste noch länger bei 1.3 bleiben.

Dass es gute Gründe geben mag, die Haupt- oder Unterversion nicht zu wechseln, ist kein Argument dafür, dass Sie innerhalb Ihrer Version nicht jeweils die aktuellste Release installieren sollten, sobald sie verfügbar ist. Dies ist aus Sicherheitsgründen im Grunde unabdingbar, weil jede neue Release alle bisher bekannt gewordenen Bugs der Vorversion beseitigt. Da die allermeisten Angriffe auf Server durch die Ausnutzung vorhandener Sicherheitslücken stattfinden, gibt es kaum eine wichtigere Sicherungsmaßnahme, die ein verantwortungsbewusster Administrator regelmäßig durchführen sollte.

Wenn Sie dieses Buch also schon eine Weile besitzen und erst dann Apache installieren möchten, sollten Sie für einen Produktions-Server nicht einfach die Version von der beiliegenden CD-ROM verwenden. Statt dessen ist es *dringend*

zu empfehlen, dass Sie zunächst auf der Website **httpd.apache.org** nachschauen, ob keine neuere Version verfügbar ist. Und selbst wenn Sie den Server nur zum Ausprobieren oder Lernen verwenden, sollten Sie diesen Rat beherzigen, denn wenn ein Angreifer erst einmal einen Zugang zu Ihrem Rechner gefunden hat, ist es in der Regel nicht der Webserver, der ihn wirklich interessiert. Näheres zu diesem ernsten und wichtigen Thema lesen Sie in Kapitel 15, *Weitere Features*.

### Funktionsübersicht

Das HTTP-Schema mit Anfrage und Antwort wurde bereits im vorigen Kapitel genau untersucht. Als Server ist Apache 2 für den Antwort-Teil (Response) zuständig: Clients (Webbrowser, Suchmaschinen-Robots und andere Programme, die Informationen aus dem Web beschaffen) senden Anfragen an den HTTP-Server. Dieser zerlegt die Anfragen in die beiden wesentlichen Bestandteile URL und zusätzliche Header. Anschließend führt er die Anfrage entweder aus oder generiert eine Meldung, die beschreibt, warum dies nicht möglich ist. In jedem Fall wird eine Antwort an den Client gesendet, die dem ebenfalls im vorigen Kapitel beschriebenen HTTP-Antwort-Schema entspricht.

So weit der grobe Ablauf. Das Interessante an einem ausgewachsenen, flexiblen und erweiterbaren Webserver wie Apache ist allerdings, dass zwischen Anfrage und Antwort beinahe beliebige Manipulationen möglich sind. Hier nur eine kleine Auswahl:

- Mit Hilfe spezieller Module (zum Beispiel `mod_alias` oder `mod_rewrite`) kann der Server den URL der Anfrage nach einem bestimmten Schema umwandeln. Dies ist beispielsweise für automatische Weiterleitungen nach Umstrukturierungen einer Website oder für intelligente halbautomatische Links interessant.
- Seit der Einführung von Apache 2 können Sie ein beliebiges externes Programm bestimmen, das die Anfrage verändern soll (Eingabefilter) oder »Last-Minute-Änderungen« an der Antwort vornimmt (Ausgabefilter). Es kann sich dabei zum Beispiel um die automatische Konvertierung von Zeichensätzen, die Herausfilterung potentiell gefährlicher Anfrageteile oder gar gezielte Textersetzungen handeln. Ausgabefilter sind erheblich leistungsfähiger als die altbekannten Server Side Includes (SSI), die Apache natürlich auch unterstützt.
- Der HTTP-Server kann auf intelligente Art und Weise entscheiden, welchen MIME-Type (Antwort-Header `Content-Type`) er für bestimmte angeforderte Ressourcen zurückliefern soll. Unter anderem kann er sich dazu an der Dateiendung oder auch am Inhalt der entsprechenden Dateien orientieren.

- Um den Bedürfnissen verschiedener Clients in optimaler Weise zu entsprechen, unterstützt Apache eine Technik, die als **Content Negotiation** bezeichnet wird: Gemäß den verschiedenen `Accept-*`-Anfrage-Headern, die im vorigen Kapitel besprochen wurden, kann der Webserver automatisch unterschiedliche Fassungen einer Ressource liefern. Einfaches Beispiel: Wenn Sie Dateien in einer Website mit einer zusätzlichen Endung abspeichern, die einem ISO-Sprachcode entspricht, wertet Apache den Header `Accept-Language` der Anfrage aus und liefert etwa eine der Dateien `index.html.de` (Deutsch), `index.html.en` (Englisch) oder `index.html.fr` aus, obwohl der entsprechende Hyperlink nur auf `index.html` lautete.
- Apache kann die Berechtigungen für Client-Anfragen auf zahlreiche Arten einschränken und diese Einschränkungen für Sie verwalten. Neben einer Autorisierung auf der Grundlage von IP-Adresse oder Hostname (`mod_access`) und der einfachen Anmeldung mit Benutzername und Passwort (`mod_auth`) wurden viele weitere interessante Authentifizierungsmodule entwickelt; beispielsweise ermöglicht `mod_auth_ldap` die Anmeldung mit Hilfe LDAP-basierter Verzeichnisdienste.
- Wie bereits angesprochen, muss die vom Client angeforderte Ressource kein statisches Dokument sein, sondern es kann sich auch um ein Skript oder Programm handeln, das die Ausgabe für den Client aus einer Vorlage und dynamischen Daten erzeugt. Apache unterstützt neben der klassischen CGI-Schnittstelle die direkte Zusammenarbeit mit zahlreichen Sprachen, Technologien und Application-Servern. Einige Beispiele, die auch in diesem Buch behandelt werden, sind die Perl-Integration mit `mod_perl`, PHP oder Java ServerPages und Java Servlets mit Tomcat.
- Sie können Apache für die abgesicherte HTTPS-Kommunikation konfigurieren: Alle zwischen Client und Server ausgetauschten Daten werden dadurch verschlüsselt und digital signiert. Dies ist für Anwendungen wie E-Commerce oder Homebanking äußerst wichtig, weil dabei private Daten über das öffentliche Internet übertragen werden, deren Integrität und Geheimhaltung gewährleistet sein muss.
- Es besteht die Möglichkeit, die meisten der im vorigen Kapitel angesprochenen Antwort-Header nachträglich zu ändern. Auch dies verbessert die Flexibilität in der Reaktion auf unterschiedliche Anfragen.
- Apache macht es Ihnen leicht, virtuelle Hosts einzurichten – mehrere vermeintlich eigenständige Websites innerhalb einer Server-Installation. Deshalb ist er der ideale Webserver für große Unternehmen mit mehreren Sites und natürlich für Webhoster, die die Websites ihrer Kunden jeweils zu hunderten gebündelt auf die Server-Maschinen packen. Für Letztere steht expli-

zit das Modul `mod_vhost_alias` zur Verfügung, das Verzeichnisse im Dateisystem nach bestimmten Vorgaben automatisch als virtuelle Hosts einrichtet.

- Wenn Sie möchten, kann sich Apache 2 sogar um andere Server-Dienste kümmern: Es gibt Module für Apache als Proxy-Server, FTP-Server, Echo-Server und sogar Mailserver. Diese Server-Dienste erledigt Apache ehrlich gesagt nicht ganz so leistungsfähig wie entsprechende Spezial-Server – aber wenn Sie sie nur gelegentlich benötigen, reicht das Leistungsspektrum völlig aus.

### Die Apache Portable Runtime

Eine der wichtigsten Neuerungen, die mit Version 2 eingeführt wurden, ist die Verwendung einer »Plattform-Abstraktionsschicht«. Dies verbessert die Stabilität und Performance des Servers insbesondere auf Nicht-UNIX-Systemen. Alle früheren Versionen verwendeten unter UNIX für den Zugriff auf Speicher, Dateien, Netzwerkfunktionen und so weiter die standardisierten POSIX-Systemaufrufe. Auf Plattformen wie Windows wurde deshalb eine nicht ganz so leistungsfähige POSIX-Emulation verwendet, auf die die Apache-Funktionalität aufsetzen konnte.

Die Apache Portable Runtime verbessert die Situation erheblich: Sie setzt auf die nativen Funktionen des jeweiligen Betriebssystems auf, statt umständlich POSIX zu emulieren, und bietet so eine plattformneutrale API für den eigentlichen HTTP-Server und seine Erweiterungen. Die Aufgabe der APR ist also vergleichbar mit derjenigen der virtuellen Maschine für Java-Programmierer oder mit der DirectX-API für die Windows-Spieleprogrammierung.

Die Entwicklung der APR ist die logische Fortsetzung einer Entwicklung, die bei Apache 1.3 begonnen hatte: Die API dieser Version enthielt bereits zahlreiche Funktionen mit dem Präfix `ap_`, also verallgemeinerte Apache-Grundfunktionen.

Die Apache Portable Runtime abstrahiert vor allem die folgenden Funktionen von Betriebssystem und Bibliotheken:

- Dateizugriffe,
- Zeichensatzkonvertierung,
- Netzwerk-Sockets,
- Datums- und Uhrzeitfunktionen,
- Text- und Zeichenkettenbehandlung,
- UNIX-artige Passwortverwaltung,

- Tabellendatenverwaltung,
- Erzeugung von UUIDs (weltweit einmaligen ID-Nummern),
- Angleichung von Datei- und Pfadnamen in ein dateisystemunabhängiges Format,
- Zufallsgenerator,
- Sperrverfahren für Dateien, Prozesse und so weiter,
- Thread- und Prozessverwaltung,
- Laden dynamischer Bibliotheken,
- Speicherverwaltung.

Wie Sie an der Liste erkennen können, gehen die Aufgaben der Apache Portable Runtime inzwischen weit über die Bereitstellung von POSIX-Alternativen hinaus. Es bot sich einfach an, nach und nach alle verallgemeinernswerten Funktionen in diese Schicht auszulagern – es entspricht dem Paradigma des modernen Software-Engineerings, dem ein Vorzeigeprojekt wie der Apache-Webserver genügt, das Rad niemals zweimal zu erfinden und alles so weit wie möglich zu modularisieren und zu verallgemeinern.

Die APR hat für für Programmierer ebenfalls einiges zu bieten; sie dient nämlich verständlicherweise nicht nur zur Implementierung der Kernfunktionalität des HTTP-Servers, sondern ermöglicht auch die Programmierung von Modulen, die genauso plattformübergreifend einsetzbar sind wie der Server selbst.

Inzwischen wird die Apache Portable Runtime übrigens nicht mehr nur für den Apache-HTTP-Server als plattformneutrale Grundlage eingesetzt, sondern auch für einige andere Open-Source-Projekte. Die APR-Website (**apr.apache.org**) listet zurzeit folgende Projekte auf:

- **Apache Flood** – ein Performance-Testprogramm für HTTP-Server, siehe **httpd.apache.org/test/flood**,
- **JXTA-C** – eine offene Peer-to-Peer-Plattform, siehe **www.jxta.org**,
- die **Tomcat-Module** `mod_jkv2` und `mod_webapp`, siehe **jakarta.apache.org/tomcat**,
- **Subversion** – eine moderne Alternative zu CVS, realisiert als Apache-Modul,
- **OPENdj** – ein Streaming-Server, siehe **www.opendj.org**,
- **mod_spin** – ein Drittanbieter-Apache-Modul mit diversen Template- und Tracking-Fähigkeiten, siehe **www.rexursive.com/software/modspin**.

Darüber hinaus verwendet der kommerzielle Softwarehersteller **Covalent** die APR in seinen Enterprise-Erweiterungen für Apache.

### Multiprocessing-Module

Bevor weiter unten die verschiedenen Multiprocessing-Module (MPMs) für Apache 2 vorgestellt werden, erhalten Sie hier eine kurze Information über das Problem, das diese Spezialmodule lösen. Dazu ist es wichtig, dass Sie sich die Aufgabe eines Netzwerk-Servers vor Augen führen: Der Server lauscht an einem bestimmten TCP-Port auf eingehende Verbindungen. Sobald eine Verbindungsanforderung ankommt, stellt er die Verbindung mit Hilfe des TCP-Drei-Wege-Handshakes her und verarbeitet die Client-Anfrage. Da lauschende TCP-Sockets so beschaffen sind, dass sie eine gewisse Anzahl wartender Client-Anfragen puffern können, würde der einfachste denkbare Fall eines TCP-Servers wie ein Supermarktkassierer einfach eine Anfrage bearbeiten, anschließend die Verbindung wieder beenden und erst dann die nächste Anforderung entgegennehmen.

In der Praxis ist dieses Verfahren allerdings absolut nicht zu empfehlen: Die Verarbeitung jeder einzelnen Anfrage kann unter Umständen so lange dauern, dass es noch vor dem TCP-Verbindungsaufbau zum Timeout kommt. Bei einem HTTP/1.0-Server wäre dies nicht das größte Problem, weil er keine persistenten Verbindungen zulassen muss. Bei vielen anderen TCP-Servern wie HTTP/1.1, FTP, Telnet oder den diversen Mail-Protokollen sind länger dauernde Verbindungen dagegen üblich. Abgesehen davon würden die Ressourcen des Server-Rechners auf diese Weise fast ununterbrochen brachliegen, weil die Geschwindigkeit durch das Netzwerk-I/O bestimmt würde. Dieses ist aber selbst bei der breitbandigsten Internet- oder LAN-Verbindung immer langsamer als die eigentliche Rechengeschwindigkeit des Computers.

Der langen Rede kurzer Sinn: Es muss eine Möglichkeit her, wie der Server mehrere Verbindungsanforderungen annehmen und parallel verarbeiten kann. Dies ist ein Problem der **Gleichzeitigkeit** oder **Nebenläufigkeit** (Concurrency). Dafür wurden im Laufe der Jahre zahlreiche unterschiedliche Lösungen entwickelt. Die wichtigsten sind folgende:

- **Forking-Server.** Der Name dieses Server-Modells stammt von dem POSIX-Systemaufruf `fork()`. Dies ist unter UNIX die übliche Methode, einen neuen Prozess zu erzeugen: Vom ursprüngliche Prozess – Parent-Prozess genannt – wird eine völlig identische Kopie angefertigt (der Child-Prozess), der alle Variablen und geöffneten Dateideskriptoren des Parent-Prozesses erbt. Beachten Sie allerdings, dass es sich aus Programmierersicht zwar um Variablen mit denselben Namen und Anfangswerten handelt, dass Änderungen in einem der beiden Prozesse aber keinen Einfluss auf die Variablen des anderen Prozesses haben. Dasselbe gilt für den Fall, dass Sie in einem Prozess Dateideskriptoren schließen, die im anderen offen bleiben. Die beiden Pro-

zesse lassen sich anhand des Rückgabewertes von `fork()` unterscheiden: Im Parent-Prozess ist es die PID des neuen Child-Prozesses, im Child-Prozess 0.

Ein Forking-Server verwendet ein lauschendes TCP-Socket, das in einer Schleife immer wieder den Systemaufruf `accept()` für den Verbindungsaufbau ausführt. Sobald dieser Befehl Erfolg hat, sobald also tatsächlich eine Verbindungsanfrage eintrifft, ruft das Programm `fork()` auf. Während der Parent-Prozess weiter Verbindungen akzeptiert, kümmert sich der Child-Prozess um die eingegangene Verbindung.

Dieses Modell ist der Klassiker unter den nebenläufigen Server-Modellen. Aus Gründen der Stabilität und Performance verwendet keines der verfügbaren Apache-MPMs dieses Verfahren. Der kleine Perl-Webserver, der im vorigen Kapitel vorgestellt wurde, arbeitet allerdings genau nach dieser Methode, weil sie sich am einfachsten implementieren lässt.

- **Preforking-Server.** Ein Server, der sehr viele Anfragen pro Minute zu verarbeiten hat, kann durch das Forking bei Eingang der Anfrage leicht in die Knie gezwungen werden. Das Erzeugen neuer Prozesse ist nämlich zeit- und ressourcenaufwändig. Aus diesem Grund besteht ein verbessertes Server-Modell darin, bereits beim Start des Servers mehrere Child-Prozesse »auf Vorrat« zu erzeugen, die sich dann um die eingehenden Verbindungsanforderungen kümmern können. Durch Fine-Tuning kann die Leistungsfähigkeit eines solchen Servers noch gesteigert werden: Die Anzahl der vorab erzeugten Prozesse und die Mindestzahl freier Prozesse, die erreicht werden muss, bevor neue erzeugt werden, kann je nach Belastung des Servers angepasst werden.

  Dieses Verfahren war bis einschließlich Version 1.3 die Methode, die Apache verwendete (zumindest auf UNIX-Systemen). In Apache 2.0 lebt es als `mpm_prefork` weiter und ist unter UNIX noch immer der Standard.

- **Threading-Server.** Moderne Betriebssysteme wie die Windows-NT-Familie, Linux ab Kernel 2.4 oder Mac OS X unterstützen eine schlanke, leichtgewichtige Alternative zu den schwerfälligen Prozessen: Genau wie ein Betriebssystem mehrere Prozesse parallel ausführen kann, können innerhalb eines Prozesses mehrere Threads laufen. Im Gegensatz zu Prozessen teilen sich alle Threads in einem Prozess denselben Speicherbereich und somit auch dieselben Variablen. Aufgrund dieser »Einsparungen« lassen sich Threads erheblich schneller und ressourcenschonender erzeugen als Prozesse.

  Das einzige Problem besteht hier darin, dass die verschiedenen Threads eben nicht gegeneinander abgeschirmt sind – dies kann bei schlechter Programmierung zu seltsamen Vermischungen zwischen den einzelnen Anfragen oder sogar zu Sicherheitsproblemen führen.

Ein gewöhnlicher Threading-Server würde ansonsten dem Forking-Server entsprechen – bei Eingang einer Anfrage wird ein neuer Thread gestartet, der diese bearbeitet. Gängiger ist allerdings ein Pre-Threading-Modell, dessen Funktionsweise dem Preforking-Server ähnelt. Apache bietet diverse Pre-Threading-MPMs; unter Windows, wo Threads seit langem zur Kernfunktionalität des Betriebssystems gehören, ist dieses Modell sogar Standard.

- **select( )-Server.** Überraschenderweise gibt es auch noch ein Server-Modell, das eben **keine** Nebenläufigkeit verwendet. Benannt ist diese Server-Variante nach dem POSIX-Systemaufruf `select()`. Dieser überprüft, ob ein Dateideskriptor (oder, wie in diesem Fall, ein Netzwerk-Socket) gerade bereit zur Ein- oder Ausgabe ist. Ein `select()`-Server verwendet diesen Aufruf, um in einer Schleife nacheinander alle offenen Verbindungen abzufragen. In jedem Socket werden dann je nach dessen Zustand einige Bytes an Daten gelesen oder geschrieben; anschließend geht es mit dem nächsten weiter. Dieses Modell erspart sich den Aufwand der Prozess- oder Thread-Erzeugung völlig.

  Moderne Betriebssysteme besitzen einen alternativen Systemaufruf namens `poll()`. Dieser überprüft nicht nur einen Dateideskriptor oder ein Socket, sondern gleich eine ganze Gruppe davon. Für jedes Socket in der Gruppe gibt er einen Status zurück. Mit `poll()` lässt sich also eine noch schnellere Variante eines solchen Servers implementieren.

  Es gibt kein Apache-MPM, das eine dieser beiden Varianten verwendet.

Wenn Sie Interesse an Details der Implementierung solcher unterschiedlicher Server haben oder sich für andere Aspekte der TCP/IP-Programmierung begeistern, empfehle ich Ihnen das Buch [Stein2002].

Apache 2.0 löst das Problem der Nebenläufigkeit auf flexible Art und Weise, nämlich durch den Einsatz von Multiprocessing-Modulen (MPMs). Sie können sich also (in gewissen plattformbedingten Grenzen) aussuchen, wie der Server mehrere Anfragen beantworten soll. Im nächsten Kapitel wird beschrieben, wie Sie bei der Kompilierung das für Sie passende Modul auswählen können. Es existieren folgende MPMs:

- `prefork`. Ein klassisches Preforking-Modell; Standard unter UNIX. Beim Start erzeugt der Server durch Forking eine bestimmte (wählbare) Anzahl von Prozessen. Diese Prozesse werden nach und nach verwendet, um die eingehenden Verbindungsanforderungen zu bearbeiten. Sobald eine Mindestzahl freier Prozesse unterschritten wird, werden wieder einige neue erzeugt.
- `worker`. Dies ist ein gemischtes Prozess- und Thread-Modell. Es profitiert gleichermaßen von der Performancesteigerung durch Threads wie von der

Stabilität eines klassischen Prozessmodells: Beim Start wird durch Preforking eine bestimmte Anzahl von Prozessen erzeugt. Jeder dieser Prozesse enthält eine feste Anzahl von Threads, die für die Verarbeitung von Verbindungen zuständig sind. Der Server wird durch Forking oder durch das Beenden von Prozessen an unterschiedliche Belastungen durch Anfragen angepasst.

- `leader`. Dies ist eine experimentelle Variante von `worker`; für einen Produktions-Server sollte sie nicht verwendet werden. Bei diesem Modell wird ein Thread zum »Leader«, die anderen sind »Followers«. Dies ermöglicht eine bessere Synchronisation der Threads.
- `threadpool`. Eine weitere experimentelle Variante von `worker`, die noch nicht für den Produktiveinsatz zu empfehlen ist. Sie basiert auf der Wiederverwendung unbeschäftigter Threads.
- `perchild`. Bei diesem Modul erzeugt eine festgelegte Anzahl von Prozessen variable Mengen von Prozessen. Die Besonderheit dieses Laufzeitmodells besteht darin, dass jeder der Prozesse unter einer anderen User-ID ausgeführt werden kann. Dies erhöht die Sicherheit für Webanwendungen oder beim virtuellen Hosting. Leider funktioniert das Modell noch nicht auf allen Plattformen.
- `mpm_winnt`. Dies ist das Standardmodell für Windows NT und seine Nachfolger. Gemäß den Gegebenheiten der Windows-Plattform setzt es voll und ganz auf Threads: Ein Parent-Prozess, der die grundlegende Steuerung des Servers übernimmt (Einlesen der Konfigurationsdaten, Neustart, Beenden) erzeugt beim Start einen einzigen Child-Prozess. Dieser enthält wiederum eine variable Anzahl von Threads, die die Client-Verbindungen bearbeiten.
- `mpm_netware`. Dieses MPM ist für Apache unter Novell NetWare optimiert. Es handelt sich um ein reines Pre-Threading-Modell: Ein Parent-Thread steuert eine variable Anzahl von Child-Threads, die sich um die einzelnen Verbindungen kümmern.
- `beos`. Ein ebenfalls rein Thread-basiertes Modell für das Betriebssystem BeOS.
- `mpmt_os2`. Das Standardmodell für IBM OS/2 funktioniert ähnlich wie das `perchild`-Modell: Ein Parent-Prozess startet eine festgelegte Anzahl von Child-Prozessen, die jeweils eine variable Anzahl von Worker-Threads verwalten.

Je nachdem, welches Laufzeitmodell Sie verwenden, stehen unterschiedliche Konfigurationsoptionen zur Verfügung, die dessen genaues Verhalten bei unterschiedlich hoher Server-Belastung bestimmen. Diese Optionen werden in Kapitel 6, *Grundkonfiguration*, behandelt.

### 3.2.2 Apache-Module

Vielleicht die wichtigste und bekannteste Eigenschaft des Apache HTTP-Servers ist dessen Erweiterbarkeit durch **Module**. Jedes Modul fügt eine bestimmte zusätzliche Fähigkeit zu Apache hinzu, deren Parameter durch einige neue Anweisungen in der Apache-Konfigurationsdatei eingestellt werden können.

Einige Module werden bereits in der Grundausstattung des Servers mitgeliefert und sind standardmäßig aktiviert. Andere sind zwar ebenfalls im Grundpaket enthalten, werden aber nur auf Verlangen einkompiliert beziehungsweise (bei Binärdistributionen) eingeschaltet. Zu guter Letzt gibt es noch Unmengen von Drittanbieter-Modulen für den Apache-Webserver.

Grundsätzlich können Module auf zweierlei Art und Weise eingebunden werden: statisch einkompiliert oder als Dynamic Shared Objects (DSOs). Letzteres ist die flexiblere Lösung, weil sich dynamisch eingebundene Module nachträglich ein- und ausschalten lassen. Eine nähere Betrachtung der Vor- und Nachteile dieser beiden Varianten des Modulbetriebs finden Sie im nächsten Kapitel.

Grob betrachtet gibt es vor allem die folgenden Gruppen von Modulen:

- **Autorisierung und Authentifizierung.** Diese Module ermöglichen die Zugriffsbeschränkung auf den Webserver oder einzelne Server-Verzeichnisse.
- **Serverseitige Programmierung.** Einige der bekanntesten Apache-Module wie `mod_cgi`, `mod_perl` oder `mod_php` sind für die Einbindung von Programmiersprachen zuständig. Sie können Programme in diesen Sprachen schreiben, deren Ausgabe als Dokument an den Client ausgeliefert wird.
- **URL-Manipulation.** Mit Hilfe dieser Module können Sie den URL einer Anfrage nach einem bestimmten Schema umwandeln. Dies ermöglicht beispielsweise intelligente Linklisten oder Sitemaps.
- **Antwort-Manipulation.** Wie Sie ausführlich in Kapitel 2, *Funktionsweise von Webservern*, erfahren haben, bestehen HTTP-Antworten unter anderem aus zahlreichen Headern. Einige Module ermöglichen Ihnen das gezielte automatische Setzen oder Ändern solcher Header. In diesen Zusammenhang gehören natürlich auch Module, die sich um die Ermittlung des passenden MIME-Types für den Antwort-Body kümmern.
- **Logging.** Für verantwortungsvolle Webmaster sind aussagekräftige Logdateien sehr wichtig. Mit Hilfe einiger Module können Sie den Standard-Inhalt der Apache-Logdateien um zusätzliche Informationen erweitern oder durch externe Programme manipulieren lassen.

- **Filter.** Eines der mächtigsten neuen Features der Version 2 ist das Vor- oder Nachschalten beliebiger Filter. Diese bieten die Möglichkeit, eingehende Daten vor der Verarbeitung durch den Server zu manipulieren (Eingabefilter) beziehungsweise die Server-Antwort vor der Auslieferung an den Client zu ändern (Ausgabefilter). Interessanterweise müssen die Filter noch nicht einmal zwangsläufig als Module realisiert sein, sondern es kann sich alternativ auch um externe Programme handeln, die bestimmte Anforderungen erfüllen.
- **Multi-Protokoll-Unterstützung.** Wie bereits erwähnt, können Sie Apache nicht nur als HTTP-Server betreiben, sondern zusätzlich auch als Webcache und Proxy-Server, als FTP-Server oder für andere TCP-basierte Anwendungsprotokolle. Einige Module sorgen für die entsprechenden Erweiterungen.

In Tabelle 3.1 finden Sie eine Liste derjenigen Module, die mit dem Quellcode-Paket des Apache-HTTP-Servers 2.0 geliefert werden. In der Spalte »Automatisch aktiviert« finden Sie Informationen darüber, ob das entsprechende Modul bei der Kompilierung des Servers mit Standardoptionen mitkompiliert wird oder nicht. Diese Information ist für die im nächsten Kapitel ausführlich beschriebene Build-Konfiguration von Bedeutung: Wenn Sie ein Modul weglassen möchten, das normalerweise aktiviert ist, müssen Sie die entsprechende `disable`-Option verwenden. Umgekehrt muss ein Modul, das automatisch deaktiviert würde, mit Hilfe einer `enable`-Option ausdrücklich hinzugefügt werden.

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| `mod_access` | ja | 6 | Durch dieses Modul können Sie Zugriffe auf bestimmte Verzeichnisse einer Website anhand von Hostname und IP-Adresse des anfragenden Clients steuern. |
| `mod_actions` | ja | 12 | Dieses Modul ermöglicht die automatische Auslösung von CGI-Handlern anhand der Dateiendung beziehungsweise des MIME-Types von Dateien. |
| `mod_alias` | ja | 8 | mod_alias ist das klassische Modul für die einfache Umwandlung von URLs. Erheblich mehr Möglichkeiten bietet das neue Modul `mod_rewrite` (siehe unten). |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.0 gehören

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_asis | ja | 7 | Mit Hilfe dieses Moduls können Sie eine Datei an den Client senden, »wie sie ist«. Das bedeutet, dass Apache für eine solche Datei keine automatischen HTTP-Header generiert; diese können Sie flexibel selbst bereitstellen. |
| mod_auth | ja | 9 | Mit Hilfe der durch dieses Modul gebotenen Basic-Authentifizierung können Sie Verzeichnisse einer Website schützen: Benutzer müssen sich durch Benutzername und Passwort anmelden, um Zugriff zu erhalten. |
| mod_auth_anon | nein | 9 | Bietet anonymen Zugriff mit dem Benutzernamen anonymous und der eigenen E-Mail-Adresse als Passwort. Diese Funktionalität wurde dem bekannten Anonymous FTP nachempfunden, hat aber auf einem Webserver in den meisten Fällen keinen Sinn. |
| mod_auth_dbm | nein | 9 | Diese Variante steigert die Effizienz, wenn Ihre Site viele Verzeichnisse mit unterschiedlichen Benutzername-Passwort-Paaren enthält: Die Anmeldedaten werden nicht in einer einfachen Passwortdatei gespeichert, sondern in datenbankähnlichen DBM-Dateien. |
| mod_auth_digest | nein | 9 | Das Modul verbessert die Sicherheit der Authentifizierung, indem es statt der BASIC-Authentifizierung mit base64-kodierten Klartextdaten MD5-Digests gemäß RFC 2617 verwendet. |
| mod_auth_ldap | nein | 9 | Dieses Modul ermöglicht die Benutzerauthentifizierung anhand existierender Daten aus LDAP-Verzeichnissen. |
| mod_autoindex | ja | 8 | Das Modul stellt den Inhalt des angeforderten Verzeichnisses auf (in Grenzen) konfigurierbare Weise dar, wenn dieses Verzeichnis keine Indexseite enthält. Ideal für umfangreiche Download-Verzeichnisse, aber ein gewisses Sicherheitsrisiko, wenn es unkontrolliert aktiviert wird: Wenn ein Website-Autor den Namen der Startseite falsch schreibt oder in einem Unterverzeichnis keine verwendet, wird die Liste aller Dateien angezeigt – bei wenig sicherheitsbewussten Webmastern können dies Dateien sein, die einen externen Besucher nichts angehen! |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.0 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_bucketeer | nein | – | Dieses Modul ist ein Beispiel für einen Ausgabefilter. Er führt bei Auftreten spezieller Sonderzeichen Buckets (interne Apache-Funktionen) aus. |
| mod_cache | nein | 9 | Dieses Modul konfiguriert Apache als Web-Cache – dies ist neben der Funktion als Anwendungs-Gateway (die durch mod_proxy zur Verfügung gestellt wird) die wichtigste Aufgabe eines Proxy-Servers. |
| mod_case_filter_in | nein | – | Dieses Modul ist ein Beispiel für einen Eingabefilter: Es wandelt die Eingabe, das heißt den Body der Client-Anfrage (zum Beispiel Formulardaten), in Großbuchstaben um. |
| mod_case_filter | nein | – | Dies ist ein Beispiel für einen Ausgabefilter: Das Modul wandelt den Body der Server-Antwort vor dem Versand an den Client in Großbuchstaben um. |
| mod_cern_meta | nein | 7 | Dieses Modul implementiert META-Files des CERN HTTPd (nur aus Kompatibilitätsgründen). Besser mod_headers verwenden! |
| mod_cgi | ja | 12 | Das Modul mod_cgi bietet die klassische Unterstützung für CGI-Skripte: Das entsprechende Programm wird als externer Prozess gestartet. Es enthält Formulardaten als Eingabe; seine Ausgabe wird an den Client weitergeleitet. |
| mod_cgid | ja | 12 | Wenn ein MPM-Modul mit Threads verwendet wird, kann mod_cgid die Performance von CGI-Skripten verbessern, da sie als leichtgewichtige Threads ausgeführt werden. |
| mod_charset_lite | ja | 14 | Das Modul bietet Unterstützung für diverse nichtlateinische Zeichensätze und kann die entsprechende Konvertierung in bestimmten Fällen automatisch vornehmen. |
| mod_dav | nein | 15 | mod_dav richtet Apache 2 als WebDAV-Server ein: Mehrere Autoren können auf den Server wie auf ein CVS-Repository zugreifen und ihre Änderungen mit Versionsinformationen versehen. |
| mod_dav_fs | nein | 15 | Bei installiertem mod_dav ermöglicht dieses Zusatzmodul den WebDAV-Zugriff auf das lokale Dateisystem. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.0 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_deflate | nein | 14 | Dieses Modul ermöglicht die Auslieferung deflate-komprimierter Daten gemäß RFC 2616 (siehe Kapitel 2), was bei modernen Browsern, die dies unterstützen, Netzwerkbandbreite spart. |
| mod_dir | ja | 6 | Dieses Modul erledigt zwei wichtige Aufgaben: Es definiert den Namen der Startseite und ermöglicht die automatische Verarbeitung von Verzeichnis-URLs, die nicht mit einem / enden. |
| mod_disk_cache | nein | 11 | Normalerweise entscheidet mod_cache selbst darüber, ob es die zwischengespeicherten Daten im Arbeitsspeicher oder auf der Festplatte ablegt. Es kann allerdings bei sorgfältiger Konfiguration die Performance steigern, dies festzulegen. Dieses Modul realisiert den Zwischenspeicher für mod_cache auf der Festplatte. |
| mod_echo | nein | 15 | Dieses Modul ist als Demonstration der Multiprotokoll-Fähigkeiten von Apache 2 gedacht und sollte normalerweise nicht auf einem Produktions-Server verwendet werden. Es konfiguriert Apache zusätzlich zu HTTP als Server für das ECHO-Protokoll, das die übergebenen Daten zurücksendet. |
| mod_env | ja | 12 | Das Modul ermöglicht die Manipulation einiger Server-Umgebungsvariablen für bestimmte Anfragen und Verzeichnisse. |
| mod_example | nein | 15 | Das Modul mod_example bietet keine wirkliche Funktionalität, sondern demonstriert allgemein, wie Apache-Module überhaupt funktionieren. Es sollte niemals auf Produktions-Servern eingesetzt werden! |
| mod_expires | nein | 7 | Das Modul ermöglicht das Setzen des HTTP/1.1-Headers Expires, der die Gültigkeitsdauer von Dokumenten für Browser und vor allem Proxies angibt. |
| mod_ext_filter | nein | 14 | Über dieses Modul können Sie fast beliebige externe Programme als Ein- oder Ausgabefilter definieren. Dies ermöglicht beispielsweise Zeichensatz- oder gar Inhaltskonvertierungen oder die gleichzeitige Verwendung von Server Side Includes und einer Server-Sprache wie Perl oder PHP. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.0 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_file_cache | nein | 11 | Mit Hilfe dieses Moduls kann Apache 2 bestimmte Dateien beim Start ins RAM kopieren. Dies ermöglicht den schnelleren Zugriff auf häufig verwendete Daten. |
| mod_headers | nein | 7 | Über mod_headers können Sie fast alle HTTP-Header setzen, ändern oder entfernen. Nur einige wenige Header sind davon ausgenommen, weil sie erst nach der Verarbeitung der Direktiven dieses Moduls automatisch gesetzt werden. |
| mod_imap | ja | 8 | Trotz seines Namens hat das Modul nichts mit dem E-Mail-Protokoll IMAP zu tun. Es bietet vielmehr interne Unterstützung für die selten gewordenen serverseitigen Image Maps. |
| mod_include | ja | 14 | Mit Hilfe von mod_include wird die grundsätzliche Unterstützung für Server Side Includes bereitgestellt. Diese relativ alte Technologie ermöglicht das schnelle Einfügen dynamischer Daten wie Dateien, Variablen oder Datum und Uhrzeit. |
| mod_info | nein | 8 | Dieses Modul ermöglicht die Einrichtung eines virtuellen URL, über den sich ausführliche Konfigurationsinformationen des Servers abfragen lassen. |
| mod_ldap | nein | 9 | Das Lightweight Directory Access Protocol hat sich in den letzten Jahren zum Standard für Verzeichnisdienste entwickelt; sowohl UNIX-Implementierungen wie OpenLDAP als auch Microsoft ActiveDirectory basieren darauf. Das Modul ermöglicht die grundsätzliche Zusammenarbeit von Apache mit LDAP-Diensten und stellt so beispielsweise Caching- und Connection-Pooling-Dienste über LDAP bereit. |
| mod_log_config | ja | 10 | Bereits ab Werk liefert Apache 2 recht aussagekräftige Logdateien. Mit Hilfe von mod_log_config können Sie noch allerlei Erweiterungen daran vornehmen und die Dateien durch externe Programme sortieren oder weiterverarbeiten lassen. |
| mod_logio | nein | 10 | Dieses Modul protokolliert zusätzlich zu den Standardinformationen die Anzahl der gesendeten beziehungsweise empfangenen Bytes in Logdateien. Es benötigt mod_log_config. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.0 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_mem_cache | nein | 11 | Ähnlich wie bei mod_disk_cache können Sie mit Hilfe dieses Moduls den Zwischenspeicher für mod_cache vorkonfigurieren. Hier wird er im RAM eingerichtet. |
| mod_mime | ja | 7 | Das Modul setzt den wichtigen Content-Type-Header der Server-Antwort automatisch anhand der Dateinamenerweiterung. |
| mod_mime_magic | nein | 7 | mod_mime_magic untersucht einige Bytes des Datei-Inhalts, um den MIME-Type für die Antwort sorgfältiger zu setzen als mod_mime. |
| mod_negotiation | ja | 7 | Dieses Modul ist für die verschiedenen Arten der Content-Negotiation – die automatische Lieferung unterschiedlicher Inhalte als Reaktion auf Accept-*-Anfrage-Header – zuständig. |
| mod_proxy | nein | 11 | Dieses Modul stellt die Grundfunktion für Apache als Proxy-Server zur Verfügung. Einige weitere Module richten spezielle Proxy-Dienste wie HTTP- oder FTP-Proxy ein. |
| mod_proxy_connect | nein | 11 | In Zusammenarbeit mit mod_proxy stellt dieses Modul eine Implementierung der HTTP-Methode CONNECT zur Verfügung: Der Proxy-Server überträgt die Daten gesicherter Verbindungen (SSL), ohne ihren Inhalt – den er ohnehin nicht verstehen kann – zu modifizieren. |
| mod_proxy_ftp | nein | 11 | Das Modul stellt einen FTP-Proxy-Server bereit, der zusammen mit mod_proxy funktioniert. |
| mod_proxy_http | nein | 11 | Wenn mod_proxy installiert ist, bietet dieses Modul die wichtigste Komponente eines Proxy-Servers, nämlich einen einfachen HTTP-Proxy. |
| mod_rewrite | nein | 8 | Dieses recht neue Modul ermöglicht die fast beliebige Umwandlung von URLs aus HTTP-Anfragen mit Hilfe regulärer Ausdrücke. |
| mod_setenvif | ja | 12 | Das Modul ermöglicht die Manipulation einiger Server-Umgebungsvariablen als Reaktion auf bestimmte HTTP-Anfrage-Header. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.0 gehören (Forts.)

| Modul | Automatisch aktiviert | Kapitel | Beschreibung |
|---|---|---|---|
| mod_speling | nein | 8 | Dieses Modul kümmert sich um die automatische Korrektur von Schreibfehlern in Anfrage-URLs, vor allem bezüglich Groß- und Kleinschreibung. Das Modul heißt tatsächlich mod_speling, mit einem l (eben ein Rechtschreibfehler). |
| mod_ssl | nein | 9 | mod_ssl stellt die grundsätzliche Unterstützung für gesicherte HTTP-Verbindungen über SSL beziehungsweise TLS zur Verfügung. |
| mod_status | ja | 8 | Das Modul liefert mit Hilfe des Textbrowsers Lynx und einer Kommandozeilenoption für das Programm httpd Statusinformationen über den laufenden Server. |
| mod_suexec | nein | 15 | suexec ermöglicht es, CGI-Skripte unter einer bestimmten User- und Group-ID auszuführen, was bei sorgfältiger Konfiguration die Sicherheit verbessert. |
| mod_unique_id | nein | – | Dieses Modul ermöglicht es, jeder einzelnen Client-Anfrage eine eindeutige ID zuzuweisen. Dies kann für bestimmte Aspekte der serverseitigen Programmierung und für Logging-Zwecke hilfreich sein. |
| mod_userdir | ja | 6 | Das Modul bildet die Home-Verzeichnisse der Benutzer (beziehungsweise ein konfigurierbares Unterverzeichnis davon) nach dem Schema http://servername/~username automatisch auf den Webserver ab. |
| mod_usertrack | nein | 10 | Dieses Modul ermöglicht das Session-Tracking durch Cookies. Sie müssen die Genehmigung der Besucher einholen, um es zu verwenden. Außerdem ist der Nutzen fraglich, da etliche Benutzer die Annahme von Cookies verweigern. |
| mod_vhost_alias | nein | 12 | Dies ist das wichtigste Modul für Hoster: Es ermöglicht die automatische Erzeugung von Unmengen virtueller Hosts anhand von Datei- und Verzeichnismustern. |

**Tabelle 3.1** Übersicht über alle Module, die zum Lieferumfang von Apache 2.0 gehören (Forts.)

In diesem Buch werden noch einige weitere wichtige Module besprochen. Im Gegensatz zu denjenigen aus Tabelle 3.1 lassen sie sich aber nicht durch standardisierte Konfigurationsparameter einkompilieren. Zu dieser Kategorie gehören unter anderem folgende Module:

- `mod_perl`. Wie bereits erwähnt, ist dieses mächtige Modul ein weiteres Projekt der Apache Software Foundation. Es ist nicht nur ein beschleunigter Ersatz für Perl-CGI-Skripte, sondern ermöglicht darüber hinaus die Erweiterung des Apache-Webservers durch selbst geschriebene sowie öffentlich verfügbare Perl-Module. mod_perl wird in diesem Buch in Kapitel 13 behandelt.
- `mod_php`. Dies ist laut Security Space (**www.securityspace.com/s_survey**) das beliebteste aller im Einsatz befindlichen Apache-Module. Die Programmiersprache PHP besticht durch eine einfach zu erlernende Syntax, eine leistungsfähige API und Schnittstellen zu zahlreichen Datenbanken (am bekanntesten dürfte in diesem Zusammenhang MySQL sein). In Kapitel 13 geht es unter anderem um PHP.
- `mod_python`. Die Skriptsprache Python ist nie an die Beliebtheit von Perl oder PHP herangekommen, obwohl sie ebenfalls ein Open-Source-Produkt mit einem immensen Funktionsumfang ist und für alle UNIX-Plattformen und für Windows zur Verfügung steht. Das Modul ermöglicht die Einbindung serverseitiger Programme, die in Python geschrieben sind.
- `mod_auth_mysql`. Neben den diversen Authentifizierungsmodulen, die zum Lieferumfang von Apache 2 gehören (`mod_auth`, `mod_auth_dbm`, `mod_auth_ldap` und so weiter), gibt es zusätzlich zahlreiche Drittanbieter-Module zur Benutzeranmeldung. Ein Beispiel ist das Modul `mod_auth_mysql`, das die Anmeldedaten in einer MySQL-Datenbank speichert.
- `mod_ftpd`. Dies ist ein weiteres Beispiel für die Multiprotokollunterstützung in Apache 2. Es macht Apache zusätzlich zum FTP-Server. Dies ist nützlich, wenn Ihre eigenen Websites (oder die Sites Ihrer Kunden) von ferne über FTP aktualisiert werden sollen: Es erspart Ihnen den Aufwand, zusätzlich einen externen FTP-Server wie WU-FTPD zu installieren.

Neben den wenigen hier aufgeführten Modulen finden Sie zahlreiche weitere in der zentralen Registry für Apache-Module unter **modules.apache.org**. Beachten Sie, dass die Mehrheit dieser Module noch immer nur für Apache 1.3 verfügbar ist und nicht mehr aktualisiert wird. Darüber hinaus genügen nicht alle dort angebotenen Module den hohen Performance- und Sicherheitsstandards, die Sie vom Apache-Webserver selbst erwarten können. Für einen Produktions-Server sollten Sie sich also nicht auf ein solches Modul verlassen, ohne zusätzliche Informationen oder Erfahrungsberichte einzuholen.

## 3.3 Zusammenfassung

Der Apache-HTTP-Server ist der mit Abstand erfolgreichste Webserver überhaupt. Kein Wunder, schließlich ist er nicht nur frei verfügbar, sondern auch stabil, sicher und höchst belastbar. Sein Einsatz auf vielen der beliebtesten Websites der Welt beweist dies auch in der Praxis.

Für die Verwendung dieses Servers spricht darüber hinaus, dass er auf zahlreichen Plattformen und Betriebssystemen funktioniert. Durch die Apache Portable Runtime (APR), die mit Version 2.0 eingeführt wurde, arbeitet der Server auch auf Nicht-UNIX-Systemen mit optimaler Performance. Hinzu kommen die verschiedenen Multiprocessing-Module (MPMs), die es ermöglichen, spezifische Leistungsvorteile einzelner Systeme zu nutzen.

Ein weiterer Pluspunkt sind die zahlreichen im Lieferumfang enthaltenen Module, die es ermöglichen, den Webserver optimal an individuelle Bedürfnisse anzupassen. Über dieselbe Schnittstelle lassen sich obendrein die verschiedensten Drittanbietermodule integrieren, die den Funktionsumfang von Apache 2 nochmals erweitern.

# 5 Apache in Betrieb nehmen

# 5 Apache in Betrieb nehmen

*Im Grunde bewegen nur zwei Fragen die Menschheit: Wie hat alles angefangen und wie wird alles enden?*
*– Stephen Hawking*



In diesem Kapitel wird die komplette Steuerung des Apache-HTTP-Servers behandelt: Sie erfahren zunächst, welche Optionen das ausführbare Programm bietet, um den Server zu starten, zu beenden oder neu zu starten. Anschließend erfahren Sie, wie sich der Start von Apache auf verschiedenen Plattformen beim Booten automatisieren lässt. Zu guter Letzt wird eine Minimalkonfiguration vorgestellt, mit der Sie die ordnungsgemäße Funktion des Servers überprüfen können.

## 5.1 Apache 2 starten und beenden

Nachdem Sie den Apache-Webserver nun (hoffentlich) auf die eine oder andere im vorigen Kapitel beschriebene Weise installiert haben, können Sie ihn starten. Je nach Betriebssystem und Plattform stehen dafür unterschiedliche Optionen zur Verfügung. Sie erfahren in diesem Abschnitt, wie sich der Server unter UNIX und Windows starten, beenden und neu starten lässt. Außerdem werden verschiedene Optionen des automatischen Starts beim Hochfahren besprochen.

### 5.1.1 Apache unter UNIX steuern

Auf UNIX-Systemen können Sie das ausführbare Programm `httpd` mit verschiedenen Parametern aufrufen, um den Webserver zu starten, neu zu starten, zu beenden oder Informationen über seinen Status zu erhalten. Die bevorzugte Variante ist allerdings die Verwendung des mitinstallierten Shell-Skripts `apachectl`, das eine genauere Kontrolle ermöglicht. Beide Varianten werden hier vorgestellt.

Ob der Webserver bereits ausgeführt wird, können Sie (als `root`) mit folgendem Kommando herausfinden:

```
# ps aux |grep httpd
```

Bei einer Standardinstallation mit dem MPM `prefork` sollte das Ergebnis etwa so aussehen wie in Abbildung 5.1, falls Apache läuft. Andernfalls erhalten Sie höchstens eine Zeile, die besagt, dass `grep httpd` ausgeführt wird – eben der Suchfilter des Befehls, den Sie gerade eingegeben haben.

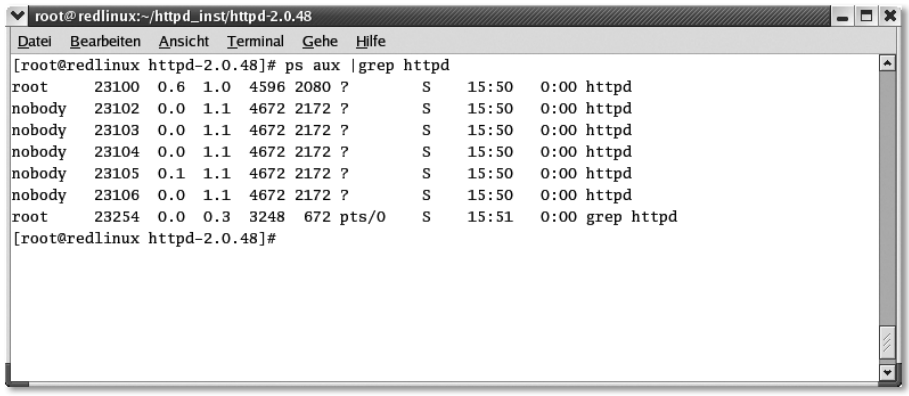



**Abbildung 5.1** Anzeige der laufenden Apache-Prozesse auf einem Linux-Host

### Parameter des Programms httpd

Der Funktionskern des Webservers Apache 2 ist das ausführbare Programm (Binary Executable) `httpd`. (Falls Sie den Server nach der Anleitung aus dem vorigen Kapitel über die Option `--with-program-name` mit einem anderen Programmnamen kompiliert haben sollten, gilt dieser im Folgenden sinngemäß.) Je nach Installationslayout beziehungsweise eingestelltem Verzeichnis befindet sich das Programm an unterschiedlichen Stellen in Ihrem Dateisystem – genauer gesagt, im `SBINDIR` Ihrer Konfiguration. Haben Sie beispielsweise das Standardlayout `Apache` verwendet, dann finden Sie die Datei im Verzeichnis `/usr/local/apache2/bin`, beim `GNU`-Layout dagegen unter `/usr/local/sbin`.

Um einen sauber installierten Apache-Webserver zu starten, sollte in der Regel der folgende Befehl ausreichen:

```
$ httpd
```

(Ob Sie dem Befehl zusätzlich eine Pfadangabe voranstellen müssen, hat natürlich damit zu tun, ob das `SBINDIR` sich in Ihrem `path` befindet oder nicht.)

Wenn der Server auf diese einfache Weise nicht startet oder wenn Sie besondere Einstellungen für seinen Start vornehmen möchten, können Sie aus den folgenden Kommandozeilenparametern auswählen:

`-D Name`

Dieser Parameter eröffnet Ihnen ein sehr praktisches Verfahren, in der Konfigurationsdatei `httpd.conf` optionale Direktiven unterzubringen und nach Belieben auszuführen oder es zu lassen: Sie können Direktiven in einen Block nach dem Schema `<IfDefine Name>...</IfDefine>` einschließen.

Sie werden nur dann ausgeführt, wenn der entsprechende `Name` beim Start von Apache mit diesem Parameter definiert wird. Die Verwendung von `<IfDefine>` in der Konfigurationsdatei wird im nächsten Kapitel erläutert.

`-d Verzeichnis`

Mit Hilfe dieses Parameters lässt sich eine alternative `ServerRoot` angeben – es handelt sich um das Verzeichnis, in dem die Inhalts- und Konfigurationsdateien des Servers liegen.

`-f Datei`

Um Ihren Server mit einer anderen Konfigurationsdatei (`ServerConfigFile`) auszuführen als mit `httpd.conf` in Ihrem `SYSCONFDIR`, können Sie diese alternative Datei mit Hilfe der vorliegenden Option angeben.

`-C "Direktive"`

Diese Option ermöglicht die Angabe einer zusätzlichen Konfigurationsdirektive, die vor dem Einlesen der Konfigurationsdatei verarbeitet werden soll.

`-c "Direktive"`

Dieser Parameter ähnelt dem vorigen, mit einem Unterschied: Eine Direktive, die Sie hier angeben, wird erst nach der Verarbeitung der Konfigurationsdatei ausgeführt – dies ermöglicht das gezielte Überschreiben einer Direktive der Server-Konfiguration.

`-e Level`

Der Parameter gibt die Dringlichkeit beim Start auftretender Fehler an, ab der diese Fehler angezeigt werden sollen. Die möglichen Werte dieser Option sind die bekannten `syslog`-Fehlerprioritäten wie `notice`, `err` oder `alert`. Sie werden weiter unten im Zusammenhang mit der Konfigurationsdirektive `LogLevel` erläutert.

`-E Datei`

Bestimmt, dass Startfehler nicht auf der Konsole (beziehungsweise `stderr`) angezeigt, sondern in die angegebene Datei geschrieben werden sollen.

Die Verwendung der folgenden Optionen startet den Server nicht, sondern dient der Ausgabe verschiedener Informationen:

`-v`

Dieser Parameter zeigt nur Versionsinformationen an.

`-V`

Zeigt ausführliche Versionsinformationen an sowie die Einstellungen, mit denen Apache 2 kompiliert wurde. Die Ausgabe sieht unter RedHat Linux mit dem Layout `GNU` und der als DSOs kompilierten Modulliste `most` beispielsweise so aus:

```
Server version: Apache/2.0.48
Server built:   Jan 24 2004 15:10:15
Server's Module Magic Number: 20020903:4
Architecture:   32-bit
Server compiled with....
 -D APACHE_MPM_DIR="server/mpm/prefork"
 -D APR_HAS_SENDFILE
 -D APR_HAS_MMAP
 -D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
 -D APR_USE_SYSVSEM_SERIALIZE
 -D APR_USE_PTHREAD_SERIALIZE
 -D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
 -D APR_HAS_OTHER_CHILD
 -D AP_HAVE_RELIABLE_PIPED_LOGS
 -D HTTPD_ROOT="/usr/local"
 -D SUEXEC_BIN="/usr/local/bin/suexec"
 -D DEFAULT_PIDLOG="var/apache2/run/httpd.pid"
 -D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
 -D DEFAULT_LOCKFILE="var/apache2/run/accept.lock"
 -D DEFAULT_ERRORLOG="logs/error_log"
 -D AP_TYPES_CONFIG_FILE="etc/apache2/mime.types"
 -D SERVER_CONFIG_FILE="etc/apache2/httpd.conf"
```

Die ersten beiden Zeilen bilden übrigens die Ausgabe der Option `-v`.

Noch ein Wort zur »Module Magic Number«: Dieser Wert entspricht der Build-Nummer der vorliegenden Apache-Release. Er ist in der Quellcode-Datei `include/ap_mmn.h` durch die symbolischen `MODULE_MAGIC_NUMBER_MAJOR` und `MODULE_MAGIC_NUMBER_MINOR` definiert. Die aktuellen Werte für Apache 2.0.49 sind `20020903` und `7`. DSO-Module mit derselben Magic Number (relevant ist die Zahl vor dem Doppelpunkt) können auf derselben Plattform im Binärformat installiert werden und brauchen nicht neu kompiliert zu werden.

`-h`

Zeigt eine Liste sämtlicher Kommandozeilenoptionen an (wie diese hier).

`-l`

Der Parameter zeigt eine Liste aller einkompilierten Module an.

`-L`

Mit dieser Option erhalten Sie eine Liste sämtlicher verfügbarer Konfigurationsdirektiven – welche das sind, hängt von den vorhandenen Modulen ab.

`-t -D DUMP_VHOSTS`

Die Option `-t -D Name` soll allgemein bestimmte Einstellungen anzeigen, die sich aus der bereits verarbeiteten Konfiguration ergeben. Zurzeit ist nur die spezielle Variante `-t -D DUMP_VHOSTS` verfügbar, die die entsprechenden Ergebnisse für virtuelle Hosts ausgibt.

`-S`

Eine Kurzfassung für die Option `-t -D DUMP_VHOSTS`.

`-t`

Diese Option überprüft die gewählte Konfigurationsdatei (und eventuelle zusätzliche Direktiven) auf syntaktische Richtigkeit.

### Apache manuell beenden und neu starten

Da in Version 2 des Apache-Webservers das weiter unten behandelte Skript `apachectl` zur Verfügung steht, ist es eigentlich nicht mehr nötig, die hier beschriebenen manuellen Verfahren anzuwenden. Allerdings verdeutlichen sie die Technik, die auch dem Shell-Skript zugrunde liegt. Deshalb ist ihre Kenntnis beispielsweise für eigene, angepasste Steuerskripte erforderlich.

Wie Sie an den oben erwähnten Parametern sehen können, enthält das ausführbare Programm `httpd` selbst keine Optionen zum Beenden oder Neustarten des Servers. Wie bei UNIX-Daemons üblich, reagiert der Steuerprozess – der ursprüngliche Parent-Prozess, der die Child-Prozesse zur Verarbeitung von Client-Verbindungen startet – statt dessen auf einige Standardsignale, die das Beenden beziehungsweise den Neustart bewirken. Solche Signale werden durch das Hilfsprogramm `kill` versandt, das den gleichnamigen Systemaufruf ausführt. Die allgemeine Syntax dieses Befehls lautet bekanntermaßen so:

```
# kill [-SIGNAL] PID
```

Das einzige Problem besteht darin, den Steuerprozess ausfindig zu machen. Wie Sie weiter oben in Abbildung 5.1 gesehen haben, gibt es bei einem `prefork`-Apache mehrere Prozesse mit dem Namen `httpd`. Welcher von ihnen der Steuerprozess ist, lässt sich auf zweierlei Arten ermitteln:

- Es ist derjenige Prozess, der unter der User-ID `root` ausgeführt wird. Die Child-Prozesse für die Client-Bedienung laufen unter einer anderen UID, meistens `nobody`.
- Die zweite Methode ist die sicherste: Apache legt beim Start eine PID-Datei an. Diese enthält die Prozess-ID des Steuerprozesses. Diese Datei wird unter anderem dazu verwendet, um bei einem Startversuch festzustellen, ob der Server bereits läuft oder ob er womöglich unsauber beendet wurde – Letzteres ist der Fall, wenn die PID-Datei zwar noch existiert, aber kein Prozess unter der Nummer läuft, die darin steht.

  Wo sich die PID-Datei (`httpd.pid`) befindet, hängt wieder einmal vom verwendeten Verzeichnislayout ab. Sie liegt im `RUNTIMEDIR`: Beim `Apache`-Layout ist dies `/usr/local/apache2/logs`, beim `GNU`-Layout `/usr/local/var/apache2/run`.

  Bequemerweise können Sie zur Angabe der PID im `kill`-Befehl unmittelbar den Wert aus `httpd.pid` verwenden. Setzen Sie dazu einfach den Ausgabebefehl `cat /Pfad/von/httpd.pid` in `` `Backticks` ``; unter Verwendung des GNU-Layouts zum Beispiel so:

  ```
  `cat /usr/local/var/apache2/run/httpd.pid`
  ```

Nun brauchen Sie nur noch zu wissen, welche Signale verwendet werden, um die unterschiedlichen Verhaltensweisen des Servers zu bewirken:

- `TERM` beendet Apache 2 vollständig. Da `TERM` das Standardsignal für `kill` ist, brauchen Sie gar kein Signal anzugeben, um Apache zu beenden.

  Hier ein Beispiel mit der nach dem ersten Verfahren ermittelten PID aus Abbildung 5.1:

  ```
  # kill 23100
  ```

  Natürlich können Sie das Signal `TERM` auch explizit angeben, wenn Sie möchten:

  ```
  # kill -TERM 23100
  ```

- `HUP` bewirkt einen normalen Neustart des Webservers: Alle Child-Prozesse werden sofort beendet (wobei laufende Übertragungen natürlich abgebrochen werden); anschließend wird der Parent-Prozess neu gestartet und erzeugt wieder die vorgesehene Anfangszahl von Child-Prozessen.

  Das folgende Beispiel funktioniert, wenn Ihr Apache 2 das `GNU`-Layout verwendet:

  ```
  # kill -HUP `cat /usr/local/var/apache2/run/httpd.pid`
  ```

- USR1 sorgt für einen unterbrechungsfreien Neustart (graceful restart) des Servers: Child-Prozesse, die sich gerade um Verbindungen kümmern, werden erst beendet, wenn der aktuelle Vorgang abgeschlossen ist; untätige Child-Prozesse werden sofort beendet. Der Parent-Prozess wird schließlich neu gestartet, nachdem alle Child-Prozesse sauber abgeschlossen wurden.

  Hier sehen Sie ein Beispiel für das Verzeichnislayout Apache:

  ```
  # kill -USR1 `/usr/local/apache2/logs/httpd.pid`
  ```



### Parameter des Skripts apachectl

Eine bequemere Steuerung des Apache-Webservers bietet das Shell-Skript apachectl, das bei der Kompilierung oder Binärinstallation von Version 2 mitgeliefert wird. Es enthält zahlreiche Optionen zum Starten, Beenden, Neustarten und Überwachen des Servers.

Das Skript befindet sich im BINDIR Ihres Verzeichnislayouts. Dies ist beim Apache-Layout /usr/local/apache2/bin, beim GNU-Layout /usr/local/bin. Die allgemeine Syntax lautet folgendermaßen:

```
# apachectl Befehl
```

apachectl bietet folgende Befehle an:

```
start
-k start
```

Der Befehl start beziehungsweise -k start startet den Webserver mit Standardoptionen. Dies ist der Standardbefehl, wenn Sie apachectl ohne weitere Optionen aufrufen.

```
stop
-k stop
```

Wenn Sie einen dieser beiden Befehle eingeben, wird der Server beendet.

```
restart
-k restart
```

Diese Befehle senden nach dem oben beschriebenen Schema ein HUP-Signal an den HTTP-Server, um ihn ohne Rücksicht auf bestehende Verbindungen sofort neu zu starten.

```
graceful
-k graceful
```

Diese beiden Befehle sorgen für einen unterbrechungsfreien Neustart (graceful restart) des Servers, der keine offenen Verbindungen fallen lässt.

`fullstatus`

Wenn in Ihrem Webserver das Modul `mod_status` aktiv ist und der Text-Browser Lynx auf Ihrem System zur Verfügung steht, gibt dieser Befehl eine Statusmeldung des laufenden Servers inklusive aller zurzeit bedienten Client-Verbindungen aus.

`status`

Auch dieser Befehl funktioniert nur, wenn `mod_status` eingeschaltet und Lynx verfügbar ist. Er gibt eine einfache Statusmeldung ohne Verbindungsinformationen aus.

`configtest`

Dieser Befehl entspricht der Option `-t` des Programms `httpd`: Er testet die syntaktische Richtigkeit der aktuellen Konfigurationsdatei.

Neben den hier beschriebenen Befehlen können Sie auch für `apachectl` alle oben beschriebenen `httpd`-Optionen verwenden; diese werden entsprechend weitergereicht.

### Den Start automatisieren

Jedes moderne UNIX-System verfügt über eine Möglichkeit, beim Booten beliebige Programme – insbesondere Daemons wie den Apache-Webserver – zu starten. Ein kleines Problem besteht nur darin, dass dieses Verfahren in den verschiedenen Systemvarianten unterschiedlich realisiert wurde. Historisch betrachtet lässt sich der Unterschied auf die beiden UNIX-Grundströmungen System V und BSD zurückführen, inzwischen zieht er sich aber – und dann auch noch mit einigen Variationen bezüglich der Verzeichniswahl – quer durch die Systeme und Distributionen (zumal es immer schwieriger wird, zu unterscheiden, welche aktuellen Systeme von System V abstammen und welche von der BSD).

Hier wird zunächst jedes der beiden grundsätzlichen Verfahren kurz vorgestellt; anschließend erhalten Sie Informationen darüber, wie sich der Server unter einem entsprechend beschaffenen System automatisch starten lässt.

- **System V Init**. Diese Boot-Methode wird von immer mehr Betriebssystemen der UNIX-Familie verwendet, unter anderem auch von Linux. Systeme, die System V Init einsetzen, arbeiten mit unterschiedlichen **Runlevels**. Ein Runlevel ist ein Systemzustand, in dem jeweils nur bestimmte Prozesse laufen. Beim Wechsel des Runlevels über den Befehl `init LEVELNR.` werden bestimmte Skripte aufgerufen, die manche Programme starten und andere beenden. Einige Runlevel haben eine spezielle Bedeutung:

- **0** Heruntergefahrener Zustand
- **S** (manchmal auch 1): Single-User-Modus (für Wartungsarbeiten)
- **1** (bei vielen Systemen): Multi-User-Modus ohne Netzwerk
- **2** Multi-User-Modus mit Netzwerk; nur Konsole
- **3** Multi-User-Modus mit Netzwerk und GUI (klassisch)
- **5** Multi-User-Modus mit Netzwerk und GUI (Linux)
- **6** Systemneustart (Reboot)



Betriebssysteme dieser Bauart besitzen für jedes Runlevel ein spezielles Init-Verzeichnis. Diese Verzeichnisse heißen `/etc/rcLEVELNR.d`, also etwa `/etc/rc1.d` für Runlevel 1 oder `/etc/rc5.d` für Runlevel 5. Die Shell-Skripte in diesen Verzeichnissen werden bei Erreichen des entsprechenden Levels automatisch ausgeführt, und zwar in alphabetischer Reihenfolge. Deshalb verwendet die übliche Konvention Namen, die mit **K** beginnen, für Kill-Skripte (die einen Prozess beenden) und solche mit **S** für Start-Skripte. Darüber hinaus bauen viele Daemons aufeinander auf. Deshalb wird hinter dem Anfangsbuchstaben eine zweistellige Zahl verwendet, die für eine bestimmte Reihenfolge sorgt.

In aller Regel sind die Einträge in diesen Verzeichnissen lediglich Symlinks auf Skripte, die sich eigentlich in einem anderen Verzeichnis befinden; meist in `/etc/init.d` oder `/sbin/init.d`. Für den Start und das Beenden des jeweiligen Prozesses wird normalerweise dasselbe Skript verwendet: Ein `S`-Symlink ruft es automatisch mit der Kommandozeilenoption `start` auf, ein `K` mit `stop`. Wie Sie weiter oben bereits erfahren haben, erfüllt das Shell-Skript `apachectl` demzufolge die Voraussetzungen für diese Aufgabe.

Aus diesem Grund brauchen Sie lediglich für Ihr Standard-Runlevel (3 oder 5) einen `S`-Symlink auf dieses Skript zu erzeugen. Für die Runlevel 0 und 6 (Herunterfahren beziehungsweise Neustart) können Sie entsprechend einen `K`-Link anlegen. Da von Apache in der Regel keine anderen Dienste abhängen, können Sie ihn recht spät starten (wählen Sie einen Symlink-Namen wie `S95apache`) und ziemlich früh beenden (`K15apache` dürfte in Ordnung gehen).

Begeben Sie sich also in das jeweilige Runlevel-Init-Verzeichnis und erstellen Sie die nötigen symbolischen Links. Angenommmen, Sie haben Apache mit dem `GNU`-Layout installiert und verwenden ein Linux-System mit dem Standard-Runlevel 5. Dann müssen Sie den folgenden Befehl für den Startskript-Link eingeben:

```
# ln -s /usr/local/bin/apachectl /etc/rc5.d/S95apache
```

Als Nächstes werden die Stopp-Links für die Runlevel 0 und 6 erzeugt. Wenn Sie das Layout `Apache` verwenden, sehen die beiden Befehle so aus:

```
# ln -s /usr/local/apache2/bin/apachectl /etc/rc0.d/K15apache
# ln -s /usr/local/apache2/bin/apachectl /etc/rc6.d/K15apache
```

- **BSD-Startskript**. BSD-basierte UNIX-Systeme verwenden im Gegensatz zu der oben beschriebenen System-V-Methode einige zentrale Startskripte. Sie befinden sich in Verzeichnissen wie `/etc` oder `/etc/rc.d` und heißen `rc.boot`, `rc.local` und so weiter. Interessant ist in diesem Zusammenhang das Skript `rc.local`, das Sie nach Belieben um weitere Startbefehle erweitern können.

  Unter einem solchen Betriebssystem brauchen Sie `rc.local` also nur mit einem Texteditor zu öffnen und können dann den Aufruf von `apachectl` mit dem Parameter `start` hinzufügen. In diesem Skript wird normalerweise mit einer Fallentscheidung nach dem Schema `if [-x PFAD]` überprüft, ob das aufzurufende Programm oder Skript überhaupt existiert. An diese Konvention sollten Sie sich halten. Falls Sie also das GNU-Layout verwenden, können Sie folgende Zeilen an `rc.local` anfügen:

  ```
  # httpd starten
  if [ -x /usr/local/bin/apachectl ]; then
     echo "Starting Apache httpd..."
     /usr/local/bin/apachectl start
  fi
  ```

Es gibt hier keine Lösung, die »besser« ist – beide funktionieren in der Praxis, und bei jedem System ist es eine von beiden.

Noch leichter haben Sie es natürlich, wenn Sie den Webserver über ein Paket Ihres Systemdistributors installiert haben: Diese Installer-Befehle kümmern sich normalerweise selbst darum, Apache für den automatischen Start zu konfigurieren. Diese Einstellung können Sie bei einigen Systemen auch auf der grafischen Benutzeroberfläche vornehmen; zu diesem Zweck bieten einige Distributionen spezielle Verwaltungsprogramme. Hier nur ein paar Beispiele:

- **SuSE Linux**. Ein gutes Argument für den Einsatz von SuSE gegenüber anderen Linux-Distributionen ist schon seit langem das komfortable Installations- und Konfigurationsprogramm `yast` (Yet Another Setup Tool). Die aktuelle Version 9.0 (Professional) bietet eine Rubrik für den automatischen Start und die komfortable grafische Konfiguration von Apache. Starten Sie dazu über ein Terminalfenster oder aus dem KDE-Menü das Programm `yast`. Hier finden Sie in der Rubrik **Netzwerkdienste** das entsprechende Icon **HTTP-Server**.

Bereits in früheren Versionen von SuSE Linux stand eine andere Möglichkeit zur Verfügung, die noch immer eingesetzt werden kann – vornehmlich, wenn Sie den Server selbst kompiliert haben, denn dann steht die `yast`-Methode nicht zur Verfügung: Über den **Runlevel-Editor** können Sie einstellen, welche Programme beziehungsweise Daemons in welchem Runlevel ausgeführt werden sollen. Starten Sie dazu wiederum `yast` und wählen Sie **Runlevel-Editor** in der Kategorie **System**. Sie sollten Apache 2 in den Runlevels 3 und 5 aktivieren (siehe Abbildung 5.2).



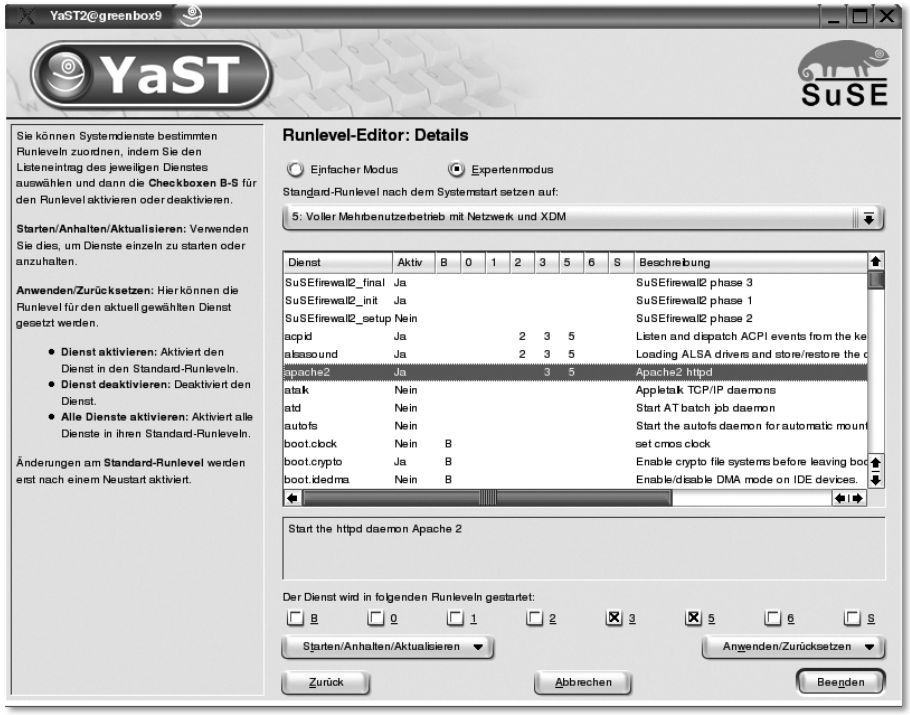



**Abbildung 5.2** Aktivieren des automatischen Apache-Starts im Runlevel-Editor von SuSE Linux

- **RedHat Linux**. RedHat enthält ebenfalls ein eigenständiges Konfigurationsprogramm für den HTTP-Server. Sie erreichen es über das RedHat-Menü in der grafischen Oberfläche unter **Servereinstellungen · HTTP-Server**. Hier können Sie zahlreiche Einstellungen für Apache vornehmen, die sich normalerweise nur manuell über die Konfigurationsdatei erreichen lassen.
- **Mac OS X**. Unter Mac OS X wird Apache normalerweise automatisch mit dem System installiert. Über **Systemeinstellungen · Netzwerk** können Sie ihn für den automatischen Start aktivieren beziehungsweise deaktivieren.

### 5.1.2 Apache unter Windows steuern

Unter Windows funktioniert die Steuerung des Webservers aufgrund der Plattformunterschiede ein wenig anders als auf UNIX-Systemen. Im Prinzip lassen sich drei verschiedene Konfigurationsarten unterscheiden, deren Funktionalität sich allerdings zum Teil überschneidet:

- Sie können das ausführbare Programm `apache.exe` mit diversen Optionen aufrufen, die zum Teil dem Programm `httpd` und dem Skript `apachectl` unter UNIX entsprechen.
- Sie können Apache 2 als Dienst installieren. Ein Dienst ist die Windows-Entsprechung eines Daemons. Wenn Apache als Dienst ausgeführt wird, können Sie ihn über das Applet **Dienste** der Systemsteuerung beziehungsweise der Microsoft Management Console steuern.
- Im Verzeichnis `bin` des Servers finden Sie ein kleines Steuerprogramm namens `ApacheMonitor.exe`. Dieses Programm installiert ein kleines Steuer-Icon in den SysTray und stellt diverse Optionen zur Verfügung.

#### Optionen des Programms Apache.exe

Unter Windows heißt das ausführbare Webserver-Programm `Apache.exe`. Es befindet sich im Verzeichnis `bin` Ihrer `ServerRoot` (zum Beispiel in `C:\Programme\Apache Group\Apache2\bin`). Im Folgenden sind alle Optionen dieses Programms aufgeführt. Sofern sie den weiter oben besprochenen Parametern von `httpd` beziehungsweise `apachectl` unter UNIX entsprechen, fällt die Beschreibung recht kurz aus:

`-D Name`

Definiert einen Namen für die Konfigurationsdirektive `<IfDefine Name>`.

`-d Verzeichnis`

Gibt eine alternative `ServerRoot` an.

`-f Datei`

Ermöglicht die Angabe einer alternativen Konfigurationsdatei.

`-C "Direktive"`

Führt vor dem Einlesen der Konfigurationsdatei die angegebene Konfigurationsdirektive aus.

`-c "Direktive"`

Führt die angegebene Direktive nach dem Verarbeiten der Konfigurationsdatei aus, ermöglicht also das nachträgliche Überschreiben vorhandener Direktiven.

`-k start`

Diese Option startet Apache. Wenn er als Dienst installiert ist, wird dieser gestartet; andernfalls startet der Server als Konsolenprogramm.

`-k runservice`

Mit diesem Befehl wird explizit ein bereits installierter Apache-Dienst gestartet.

`-k restart`

Diese Option führt einen unterbrechungsfreien Neustart des Servers durch. Wenn er als Konsolenprogramm läuft, müssen Sie ein weiteres Eingabeaufforderungsfenster öffnen, um den Befehl einzugeben.

`-k stop`
`-k shutdown`

Jeder dieser beiden Befehle beendet Apache. Läuft er als Konsolenprogramm, dann können Sie den Befehl von einem anderen Fenster aus eingeben.

`-k install`

Installiert Apache als Dienst.

`-k config`

Diese Option kann verwendet werden, um zusammen mit anderen Befehlen die Konfiguration des Apache-Dienstes zu ändern.

`-k uninstall`

Deinstalliert den Apache-Dienst.

`-n Name`

Mit Hilfe dieser Option können Sie einen alternativen Namen angeben, unter dem der Apache-Dienst installiert werden soll (der Standardname ist `Apache2`). Hat er bereits einen anderen Namen, dann dient dieselbe Option dazu, den Dienst später für die Deinstallation oder für Konfigurationsänderungen anzusprechen.

`-w`

Wenn diese Option angegeben wird, bleibt das Konsolenfenster bei einem Fehler geöffnet. Dies ist vor allem dann nützlich, wenn Sie Apache aus einer Batchdatei heraus starten, die per Doppelklick oder automatisch ausgeführt wird.

`-e level`

Dringlichkeitsstufe, ab der Fehler beim Start des Servers angezeigt werden sollen.

`-E Datei`

Schreibt Startfehlermeldungen in die angegebene Datei.

`-v`

Ausgabe von Versionsinformationen.

`-V`

Ausgabe der Versionsinformationen und der Einstellungen, mit denen der Server kompiliert wurde.

`-h`

Ausgabe einer Liste aller Kommandozeilenoptionen.

`-l`

Ausgabe einer Liste der einkompilierten Module.

`-L`

Auflisten der verfügbaren Konfigurationsdirektiven.

`-t -D DUMP_VHOSTS`

Ausgabe der verarbeiteten Einstellungen für virtuelle Hosts.

`-S`

Kurzfassung von `-t -D DUMP_VHOSTS`.

`-t`

Syntax der Konfigurationsdatei überprüfen.

### Apache als Dienst betreiben

Bereits seit der Vorgängerversion 1.3 lässt sich Apache unter NT-basierten Windows-Betriebssystemen als Dienst installieren. Der Vorteil ist, dass der Server in dieser Konstellation unabhängig von einem angemeldeten Benutzer ausgeführt wird. Auch die Performance ist bei einem Dienst besser, als wenn Apache 2 als Konsolenanwendung ausgeführt wird.

Wenn Sie das im vorigen Kapitel beschriebene Windows-MSI-Paket installieren, können Sie die automatische Einrichtung von Apache als Dienst wählen; sie ist sogar standardmäßig vorgegeben. Haben Sie den Server dagegen selbst kompiliert oder die Installation als Dienst abgelehnt, müssen Sie den folgenden Befehl ausführen, um den Dienst nachträglich zu installieren.



```
> apache -k install
```

Der Dienst wird dadurch ein für allemal in die Liste der Systemdienste aufgenommen. Der Standardname des Dienstes ist `Apache2`. Um einen anderen Namen festzulegen, können Sie den Befehl mit der zusätzlichen Option `-n Name` verwenden, beispielsweise so:

```
> apache -k install -n Winnetou
```

Wenn Sie ihn später wieder entfernen möchten, können Sie dies mit diesem Befehl erledigen:

```
> apache -k uninstall
```

Falls Sie einen alternativen Namen festgelegt haben, müssen Sie diesen auch bei der Deinstallation angeben:

```
> apache -k uninstall -n Winnetou
```

Einen installierten Apache-Dienst können Sie auf der Kommandozeile auch mit Hilfe der Befehle `net start` und `net stop` steuern. Dazu müssen Sie in jedem Fall den Dienstnamen angeben, nicht den Namen des ausführbaren Programms. Beispiele:

```
> net start Apache2
> net stop Winnetou
```

Zur Verwaltung von Diensten wie dem Apache-Dienst bietet Windows das Verwaltungs-Applet **Dienste**. In Windows XP finden Sie es unter **Start · Verwaltung · Dienste**, unter Windows 2000 in der Systemsteuerung unter **Verwaltung** und bei Windows NT 4.0 unter **Start · Einstellungen · Systemsteuerung · Dienste**. Abbildung 5.3 zeigt das Applet unter Windows 2000.

Änderungen an der Konfiguration des Dienstes können Sie über die Schaltfläche **Eigenschaften** vornehmen; es handelt sich um die Schaltfläche mit dem »Gepäckanhänger«, den Sie in der Symbolleiste in Abbildung 5.3 sehen können. Dieselbe Symbolleiste enthält im Übrigen Schaltflächen zum schnellen Starten, Beenden und Neustarten des gerade ausgewählten Dienstes.

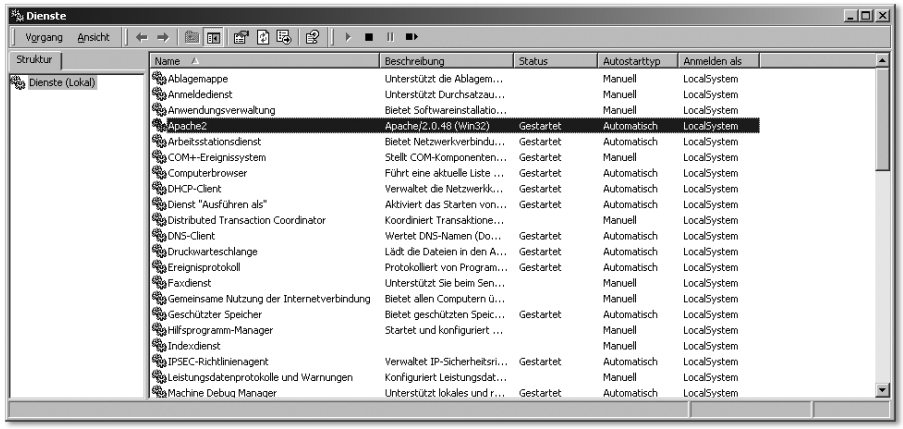



**Abbildung 5.3** Das Verwaltungs-Applet »Dienste« mit ausgewähltem Apache-Dienst unter Windows 2000

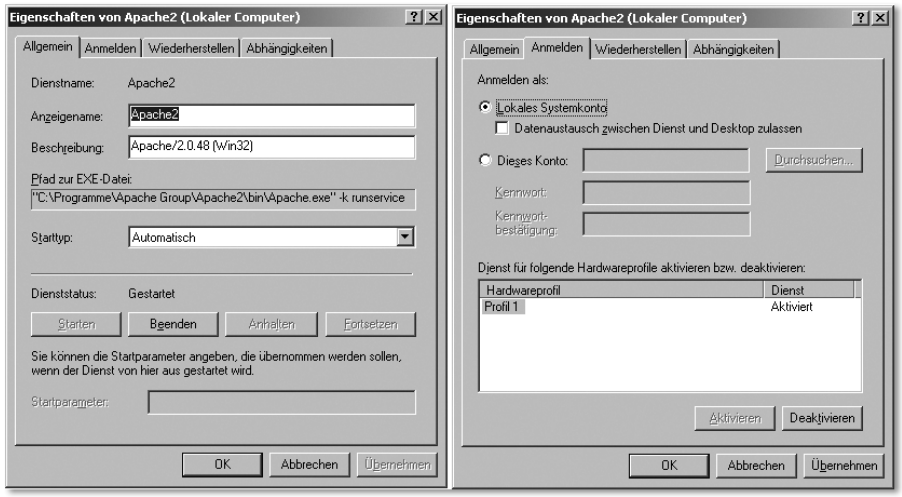



**Abbildung 5.4** Die beiden ersten (und wichtigsten) Registerkarten des Eigenschaften-Dialogs für den Apache-Dienst unter Windows 2000

In Abbildung 5.4 werden nebeneinander die Inhalte der beiden ersten Registerkarten des Eigenschaften-Dialogs für den Apache-Dienst gezeigt: **Allgemein** und **Anmelden**. Die vier Registerkarten des Dialogs bieten die folgenden Einstellungsmöglichkeiten:

- **Allgemein.** Hier werden die Grundeinstellungen für den Dienst vorgenommen, insbesondere ermöglicht der **Starttyp** die Einstellung, ob er automatisch gestartet werden soll. Im Einzelnen finden Sie hier folgende Felder und Schaltflächen:

- **Dienstname.** Offizielle Bezeichnung des Dienstes; nicht änderbar.
- **Anzeigename.** Änderbarer Name des Dienstes, wie er in der Liste angezeigt wird.
- **Beschreibung.** Nähere Information über den Dienst.
- **Pfad zur EXE-Datei.** Das ausführbare Programm, das dieser Dienst ausführt, mitsamt Kommandozeilenparametern.
- **Starttyp.** Legt fest, wie dieser Dienst gestartet werden soll: **Automatisch** startet ihn automatisch, **Manuell** nur auf ausdrückliche Anforderung und **Deaktiviert** gar nicht.
- **Dienststatus.** Zeigt an, ob der Dienst zurzeit läuft, beendet wurde oder deaktiviert ist.
- **Starten.** Hier können Sie den Apache-Dienst starten, wenn Sie die Startart auf Manuell gesetzt oder ihn zuvor beendet haben.
- **Beenden.** Diese Schaltfläche beendet den Apache-Dienst.
- **Anhalten.** Einige Windows-Server-Dienste – meist von Microsoft selbst – können über diese Schaltfläche in eine Art »Administrationsmodus« versetzt werden: Der Dienst ist dann nur noch für Benutzer mit Administratorrechten erreichbar. Bei Apache wurde dieses Feature leider noch nicht eingebaut; hier müssen Sie sich mit einer Umkonfiguration der Authentifizierung (siehe Kapitel 9, *Authentifizierung und gesicherte Verbindungen*) behelfen. Daher ist die Schaltfläche beim Apache-Dienst deaktiviert.
- **Fortsetzen.** Das Gegenstück zur Schaltfläche **Anhalten:** Ein Dienst soll aus dem eingeschränkten Wartungsmodus wieder in den Normalbetrieb zurückversetzt werden. Für Apache ebenfalls nicht verfügbar.

- **Anmelden.** Auf dieser Registerkarte wird festgelegt, unter welcher Benutzerkennung der Dienst `Apache2` ausgeführt wird. Die Standardeinstellung ist **Lokales Systemkonto**. Dies ist in den meisten Fällen in Ordnung, mit einer wichtigen Ausnahme:

  Aus Sicherheitsgründen sollte das Benutzerkonto des Apache-Dienstes niemals Berechtigungen für Netzwerkanwendungen erhalten. Falls der »Benutzer« **Lokales Systemkonto** diese für eine andere Anwendung benötigt, sollten Sie Apache unbedingt unter einer anderen Benutzerkennung betreiben. Dazu müssen Sie einen neuen Benutzer einrichten; dieser sollte keine Administratorrechte haben, sondern ein normaler Benutzer sein. Er benötigt aber die zusätzlichen Rechte »Anmelden als Dienst« und »Als Teil des Betriebssystems handeln«.

Unter Windows XP legen Sie einen neuen Benutzer über das Applet **Benutzerkonten** in der Systemsteuerung an, unter Windows 2000 unter **Verwaltung**. Die erforderlichen Rechte können Sie über Gruppenrichtlinien oder über die lokalen Sicherheitseinstellungen in der Microsoft Management Console vergeben. Bei Windows NT 4.0 erledigen Sie beides mit Hilfe des Programms unter **Start · Programme · Verwaltung (Allgemein) · Benutzer-Manager**; die Rechte finden Sie dort unter **Richtlinien · Benutzerrechte**.

Wählen Sie nach dem Erstellen des neuen Benutzerkontos die Option **Dieser Benutzer**. Geben Sie den Namen dieses Benutzers und zweimal sein Passwort ein.

Unter der Einstellung des Benutzerkontos können Sie noch festlegen, in welchen **Hardwareprofilen** Apache aktiviert werden soll. Da Sie beim Booten ein bestimmtes Hardwareprofil auswählen können, bietet dieses Verfahren eine einfache Möglichkeit, Betriebssystemkonfigurationen mit und ohne aktivierten Apache-Webserver einzurichten. Eingerichtet werden Hardwareprofile übrigens in den **Systemeigenschaften** (**Systemsteuerung · System** oder rechte Maustaste auf das Symbol **Arbeitsplatz** und **Eigenschaften** auswählen).

- **Wiederherstellen.** Auf dieser Registerkarte können Sie detailliert festlegen, was geschehen soll, wenn Apache ausfällt, das heißt beim Systemstart nicht ausgeführt werden kann oder unerwartet beendet wird. Dies ist natürlich besonders dann wichtig, wenn Sie den betreffenden Rechner hauptsächlich als Webserver für ein Intranet oder das Internet verwenden.

  Unter den Kategorien **Erster Fehlschlag**, **Zweiter Fehlschlag** und **Weitere Fehlschläge** können Sie je eine der folgenden Optionen auswählen:

  - **Keinen Vorgang durchführen.** Es soll nichts Besonderes veranlasst werden. Wenn Sie Apache nur zu Test- oder Entwicklungszwecken installiert haben, ist dies die passende Auswahl.
  - **Dienst neu starten**. Es soll versucht werden, Apache neu zu starten. Für den ersten Fehlschlag bietet sich diese Möglichkeit an.
  - **Datei ausführen**. Wenn Sie diese Option auswählen, können Sie weiter unten eine Datei bestimmen, die ausgeführt werden soll. Dies kann ein Programm sein oder eine beliebige Datei, deren Dateityp Windows einer Anwendung zuordnen kann. Zusätzlich besteht die Möglichkeit, dieser Datei die Fehlschlagnummer als Kommandozeilenparameter in der Form `/fail=%1%` zu übergeben. Ein entsprechend präpariertes eigenes Programm kann diesen Parameter auswerten.

- **Computer neu starten.** Diese extreme Option ergibt eigentlich nur dann einen Sinn, wenn Sie einen Windows-Rechner in Ihrem Netzwerk ausschließlich als Webserver einsetzen. Unter **Informationen über Neustart** können Sie in diesem Fall eine Meldung eintragen; diese wird allen (Windows-)Benutzern im Netzwerk angezeigt, die zur Zeit des Neustarts mit diesem Host verbunden sind.



- **Abhängigkeiten.** Auf dieser Registerkarte werden die Dienste angezeigt, von deren Funktionieren der aktuelle Dienst abhängt und umgekehrt. Von Apache hängen in der Regel keine anderen Dienste ab. Er selbst benötigt natürlich funktionierendes TCP/IP-Networking, was bei neueren Windows-Versionen auch als Voraussetzung angezeigt wird.

Beachten Sie, dass dieser Dialog unter Windows NT 4.0 nur aus einer einzigen Seite besteht. Auf dieser können Sie lediglich den Starttyp und das Benutzerkonto einstellen und natürlich den Dienst beenden und neu starten.

Windows *95*, *98* und *Me* bieten von Hause aus gar keine Dienste an. Aber obwohl der Einsatz von Apache unter diesen Systemen ohnehin nicht zu empfehlen ist, wurde der als Nächstes beschriebene Apache-Monitor so geschrieben, dass er auch mit einer Art »Dienstemulation« zusammenarbeitet, die auf diesen Systemen läuft.

### Der Apache-Monitor

Wenn Sie Apache über den MSI-Installer als Dienst installiert haben, wurde der Apache-Monitor automatisch eingerichtet und wird bei jedem Windows-Bootvorgang mitgestartet. Bei anderen Installationsarten können Sie ihn selbst starten oder ebenfalls für den automatischen Start konfigurieren.

Das Programm trägt den Namen `ApacheMonitor.exe` und befindet sich im Verzeichnis `bin` des Apache-Installationsordners. Wenn Sie es per Doppelklick oder über die Konsole starten, bleibt es nur für die aktuelle Systemsitzung aktiv. Um es für den automatischen Start einzurichten, gibt es zwei Möglichkeiten:

- Sie können eine Verknüpfung zu dem Programm im Ordner **Autostart** des Startmenüs anlegen. Dazu genügt es, das Icon des Programms mit der rechten Maustaste in **Start · Alle Programme · Autostart** zu ziehen und beim Loslassen die Option **Verknüpfung hier erstellen** aus dem Kontextmenü zu wählen. Diese Variante verwendet übrigens auch der MSI-Installer automatisch.

- Die Alternative besteht darin, einen Eintrag für den Start des Monitors in der Registry zu erstellen. Wählen Sie dazu **Start · Ausführen** und geben Sie `regedit` ein. Im linken Fensterbereich müssen Sie sich durch die Hierarchie zum Schlüssel **HKEY_LOCAL_MACHINE\ Software\ Microsoft\ Windows\ CurrentVersion\ Run** vorarbeiten. Wenn Sie den Ordner **Run** angeklickt haben, können Sie mit der rechten Maustaste in den rechten Fensterbereich klicken und **Neu · Zeichenfolge** aus dem Kontextmenü wählen. Geben Sie einen beliebigen Namen ein (in diesem Fall natürlich am besten **ApacheMonitor**). Doppelklicken Sie zu guter Letzt auf das Icon der neuen Zeichenfolge und geben Sie als Wert den Pfad des Programms `ApacheMonitor.exe` ein. In Abbildung 5.5 sehen Sie, wie es gemacht wird.

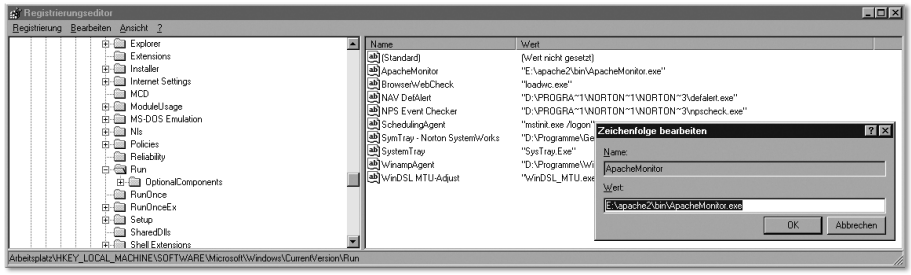



**Abbildung 5.5** Den automatischen Start von ApacheMonitor.exe in der Windows-Registry einrichten

Wenn der Apache-Monitor ausgeführt wird, macht er sich als kleines Apache-Icon im SysTray bemerkbar – dies ist der Bereich links in der Taskleiste, neben der Uhrzeit. Wenn Sie das Icon mit der linken Maustaste anklicken, stehen Ihnen die Optionen **Start** (den Apache-Dienst starten) **Stop** (zum Beenden des Dienstes) und **Restart** (Neustart, zum Beispiel nach einer Konfigurationsänderung) zur Verfügung. Ein Klick mit der rechten Maustaste ermöglicht dagegen das Öffnen des eigentlichen Monitor-Fensters, das in Abbildung 5.6 zu sehen ist.

Auch hier sind zunächst einmal wieder Schaltflächen zum Starten, Beenden und Neustarten zu erkennen. Darüber hinaus können Sie über **Connect** eine Verbindung zu einem anderen Windows-Rechner in Ihrem LAN herstellen, der Apache 2 ausführt, und dessen Apache-Dienst fernsteuern. Dazu benötigen Sie allerdings Administratorrechte, die auch für den entfernten Host gelten – entweder über ein entsprechendes Domänen-Benutzerkonto oder dadurch, dass Ihr aktueller Benutzername mit demselben Passwort und identischen Rechten auf dem anderen Computer existiert.

Die Schaltfläche **Services** schließlich öffnet unter Windows 2000 und XP das weiter oben besprochene Betriebssystem-Applet **Dienste**; unter Windows NT 4.0 funktioniert dies leider nicht.

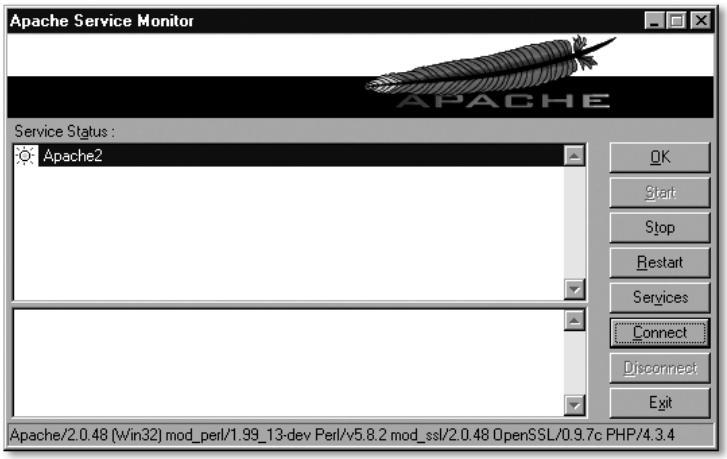



**Abbildung 5.6** Das Hauptfenster des Apache-Monitors

### 5.1.3 Apache-Hilfsprogramme

Das Programm `httpd` (UNIX) beziehungsweise `apache.exe` (Windows) ist nicht das einzige ausführbare Programm, das mit dem Webserver installiert wird. Es gibt zusätzlich einige nützliche Kommandozeilentools. Die meisten von ihnen sind vor allem für Aufgaben nützlich, auf die erst in späteren Kapiteln näher eingegangen wird. Aus diesem Grund finden Sie hier nur eine kurze Übersicht über die Programme mit dem Hinweis, in welchem Kapitel sie jeweils behandelt werden.

Im Einzelnen stehen neben dem ausführbaren Server-Programm und dem Skript `apachectl` folgende Programme zur Verfügung:

- `ab` – das Apache-Benchmark-Programm. Siehe Kapitel 11, *Skalierung, Load-Balancing und Proxies*.
- `apxs` – Hilfsprogramm zum nachträglichen Kompilieren von Modulen. Dieses Tool wurde bereits in Kapitel 4, *Apache kompilieren und installieren*, angesprochen.
- `dbmmanage` – Perl-Skript zur Verwaltung von Authentifizierungsdaten im DBM-Format. Siehe Kapitel 9, *Authentifizierung und gesicherte Verbindungen*.
- `htdbm` – Ein neueres, binäres Verwaltungsprogramm für DBM-Authentifizierungsdaten. Siehe Kapitel 9.

- `htdigest` – Verwaltungsprogramm für Digest-Authentifizierungsdaten. Näheres in Kapitel 9.
- `htpasswd` – Verwaltungsprogramm für Basic-Authentifizierungsdaten. Auch dieses Hilfsprogramm wird in Kapitel 9 behandelt.
- `logresolve` – Tool zur Ermittlung von Hostnamen zu den IP-Adressen in Logdateien. Siehe Kapitel 10, *Logging*.
- `rotatelogs` – Programm zum automatischen regelmäßigen Wechsel von Logdateien. Auch dieses Programm wird in Kapitel 10 näher betrachtet.
- `suexec` – CGI-Skripte unter anderer User-ID ausführen. Siehe Kapitel 15, *Weitere Features*.
- `log_server_status` – Statusinformationen in eine Zeile packen und in eine Logdatei schreiben. Näheres in Kapitel 10.
- `split-logfile` – Logdateien anhand bestimmter Regeln in mehrere Einzeldateien zerlegen. Ebenfalls in Kapitel 10.

## 5.2 Apache testen

Nach Installation und Start sollten Sie überprüfen, ob Apache ordnungsgemäß funktioniert. In diesem kurzen Abschnitt werden zwei Methoden dafür beschrieben: das Überprüfen der mitgelieferten Startseite und das Einrichten einer eigenen Minimalkonfiguration

### 5.2.1 Die automatische Startseite

Nachdem Sie den Webserver mit einer der hier beschriebenen Methoden gestartet haben, sollte er eigentlich funktionieren. Ob dies tatsächlich der Fall ist, können Sie mit einem Webbrowser testen: Das vorkonfigurierte Website-Verzeichnis `htdocs` enthält zu diesem Zweck eine Testseite, die im Browser angezeigt werden sollte, wenn Sie die Wurzeladresse Ihres Servers eingeben.

Öffnen Sie also einen Browser und geben Sie als URL **http://localhost** ein. Falls der Standardname `localhost` auf Ihrem System nicht unterstützt wird, müssen Sie statt dessen **http://127.0.0.1** eingeben. Abbildung 5.7 zeigt, wie die Seite aussehen sollte, wenn alles in Ordnung ist. Wenn Ihr Browser keinen `Accept-Language`-Header mit der Sprachpräferenz Deutsch sendet oder der Webserver nicht für Content Negotiation konfiguriert ist, werden Sie die Seite allerdings – anders als in der Abbildung – auf Englisch zu sehen bekommen.


**Abbildung 5.7** Die Testseite des Apache-Webservers nach erfolgreichem Start

### 5.2.2 Die erste Website

Es genügt natürlich nicht, den Apache-Webserver einfach so zu starten. Es sind zahlreiche Anpassungen der Konfigurationsdatei `httpd.conf` erforderlich, damit er genau nach Wunsch arbeitet. Da große Teile des Inhalts von `httpd.conf` davon abhängen, welche Module im Apache-Server installiert sind, werden in diesem Unterabschnitt nur einige wenige Konfigurationsdirektiven angesprochen, die das Ausliefern statischer Dokumente ermöglichen.

Wenn Sie eine neue Apache-Installation zum ersten Mal verwenden (und vor allem, wenn Sie Apache überhaupt das erste Mal einsetzen), sollten Sie nicht einfach ohne weiteres eine Website auf die Menschheit loslassen. Es gibt Unmengen von Konfigurationsanweisungen und Anpassungsmöglichkeiten. Aus diesem Grund empfiehlt es sich, einen neu installierten Server vor der Inbetriebnahme mit einer Test-Website zu überprüfen.

In diesem kurzen Abschnitt wird eine kleine Website mit einer minimalen Konfigurationsdatei erstellt. Einzelheiten zu den zahlreichen Optionen für die Datei `httpd.conf` finden Sie weiter unten im Buch; hier geht es erst einmal darum, überhaupt eine Website zu veröffentlichen.

Erstellen Sie als Erstes ein Verzeichnis, das den Stamm Ihrer Website bilden soll. Erstellen Sie darin eine HTML-Datei namens `index.html` mit beliebigem Inhalt (oder kopieren Sie den Inhalt des Verzeichnisses `testsite` von der CD zum Buch hinein). Als Verzeichnis für den Test können Sie beispielsweise das

offizielle `htdocs`-Verzeichnis Ihres Apache-Servers oder ein neu erstelltes Unterverzeichnis davon benutzen.

Benennen Sie die vorgefertigte Konfigurationsdatei `httpd.conf` um. Erstellen Sie die folgende neue Minimaldatei (Erläuterung der austauschbaren Elemente siehe unten):

```
ServerName www.mynet.de
Listen 80
ServerRoot /usr/local/apache2

# Die folgenden drei Zeilen werden nur bei
# DSO-basierter Modulinstallation benötigt:
LoadModule dir_module modules/mod_dir.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule access_module modues/mod_access.so

DocumentRoot /usr/local/apache2/htdocs

# Absicherung durch Absperren des Wurzelverzeichnisses
<Directory />
  Options None
  AllowOverride None
  Order deny,allow
  Deny from all
</Directory>

# Freigeben der DocumentRoot
<Directory /usr/local/apache2/htdocs>
  DirectoryIndex index.html
  Options All
  AllowOverride All
  Order allow,deny
  Allow from all
</Directory>
```

Die Verzeichnisangaben für die Direktiven `ServerRoot`, `DocumentRoot` und den `<Directory>`-Container müssen Sie natürlich den Gegebenheiten Ihrer Plattform und Ihrer Apache-Installation anpassen. Das obige Beispiel entspricht dem `Apache`-Layout unter UNIX.

Zu beachten ist in diesem Zusammenhang, dass auch unter Windows der Slash (/) und nicht der plattformtypische Backslash (\) als Pfadtrennzeichen benutzt

werden muss. Angenommen, Apache ist auf Ihrem Windows-Rechner unter `C:\Programme\Apache2` installiert, dann sieht die Datei folgendermaßen aus:

```
ServerName www.mynet.de
Listen 80
ServerRoot C:/Programme/Apache2

LoadModule dir_module modules/mod_dir.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule access_module modules/mod_access.so

DocumentRoot C:/Programme/Apache2/htdocs

# Absicherung durch Absperren des Wurzelverzeichnisses
<Directory />
  Options None
  AllowOverride None
  Order deny,allow
  Deny from all
</Directory>

# Freigeben der DocumentRoot
<Directory C:/Programme/Apache2/htdocs>
  DirectoryIndex index.html
  Options All
  AllowOverride All
  Order allow,deny
  Allow from all
</Directory>
```

Wie Sie sehen, sind die beiden `LoadModule`-Direktiven unter Windows nicht optional: Bei der Standardversion, die der im vorigen Kapitel vorgestellte MSI-Installer einrichtet, werden Module immer als DSOs geladen und sind niemals statisch einkompiliert.

Im Übrigen sollten Sie **www.mynet.de** für einen lokalen Test zu Ihrer `hosts`-Datei (UNIX: `/etc/hosts`, Windows: `%Systemroot%\System32\drivers\etc\hosts`) hinzufügen. Andernfalls erhalten Sie beim Apache-Start die folgende Meldung:

```
Could not qualified the Server's full qualified domain name.
Using 127.0.0.1 for Server Name.
```

Erstellen Sie in der Datei `hosts` einen Eintrag wie diesen, um den Server auf demselben Host zu testen:

```
www.mynet.de
```

Für einen Test über das Netzwerk müssen Sie statt www.mynet.de die IP-Adresse der entsprechenden Netzwerkschnittstelle verwenden. Bei einem Produktions-Server muss der Name gemäß den Erläuterungen in Kapitel 1, *IP-Netzwerke, Internet und WWW*, über einen DNS-Server aufgelöst werden, damit der Server im Internet unter diesem Namen erreichbar ist.

Starten Sie Apache 2 nun neu, indem Sie der weiter oben beschriebenen Anleitung für Ihre Plattform folgen. Falls Sie die Minimalkonfigurationsdatei unter einem anderen Namen als `httpd.conf` gespeichert haben, können Sie sie dabei mit der Option `-f Dateiname` angeben. Angenommen, Sie verwenden ein UNIX-System und die Datei `/etc/httpd/httpd_minimal.conf`, dann sieht der entsprechende Befehl so aus:

```
# apachectl graceful -f /etc/httpd/httpd_minimal.conf
```

Auf einem Windows-Rechner müssen Sie dagegen diesen Befehl eingeben, falls Sie keine der bereits behandelten grafischen Methoden einsetzen und die Konfigurationsdatei `C:\Programme\Apache2\conf\httpd_minimal.conf` verwenden:

```
> apache -k restart
  -f C:/Programme/Apache2/conf/httpd_minimal.conf
```

Nach dem Neustart können Sie einen Browser öffnen und versuchen, die neue Startseite der Website aufzurufen. Falls alles in Ordnung ist, müssten Sie nun die eben erstellte HTML-Seite sehen.

Hier noch eine kurze Übersicht über die Direktiven, die in der Minimalkonfiguration verwendet werden – sie alle werden im nächsten Kapitel genauer erläutert:

`ServerName www.mynet.de`

Hier wird der Name festgelegt, unter dem der Server im Intra- oder Internet erreichbar sein soll.

`Listen 80`

Diese Direktive legt den TCP-Port fest, an dem der Server auf HTTP-Verbindungsanfragen lauscht. Wie Sie bereits wissen, ist 80 der Standardport für das HTTP.

`ServerRoot /usr/local/apache2`

Die Konfigurationsanweisung `ServerRoot` legt das Verzeichnis fest, in dem sich die Konfigurations- und Ressourcendateien des Servers befinden, das heißt Verzeichnisse wie `conf`. Verschiedene andere Direktiven benötigen als Parameter eine Pfadangabe, die relativ zu diesem Verzeichnis angegeben werden kann.



`LoadModule dir_module modules/mod_dir.so`

Diese Zeile und die beiden nächsten werden auf einem UNIX-System nur benötigt, wenn alle Module (oder die drei Module `mod_dir`, `mod_autoindex` und `mod_access` ausdrücklich) als DSOs installiert wurden. Mit Hilfe von `LoadModule` wird ein DSO-Modul beim Server-Start geladen und aktiviert. Es werden jeweils ein schematischer Modulname und der Pfadname des Moduls (relativ zur `ServerRoot`) benötigt. Für den schematischen Namen müssen Sie das Präfix `mod_` vom Dateinamen entfernen und statt dessen `_module` anfügen. Aus `mod_foo` würde also beispielsweise `foo_module` mit folgender Direktive:

```
LoadModule foo_module modules/mod_foo.so
```

Unter Windows werden die `LoadModule`-Direktiven fast immer verwendet, weil Module auf dieser Plattform in aller Regel als DSOs installiert werden.

`mod_dir` ist übrigens das Modul, das die Definition des Startseitennamens ermöglicht und darüber hinaus Verzeichnis-URLs ohne abschließenden / verarbeitet. `mod_autoindex` generiert dagegen automatisch einen Index aller Dateien im Verzeichnis, wenn keine Startseite vorhanden ist. `mod_access` schließlich erlaubt Zugriffsbeschränkungen auf Hostnamens- und IP-Adress-Ebene.

`DocumentRoot /usr/local/apache2/htdocs`

Dies ist das Verzeichnis, in dem sich die Website befindet, die der Server veröffentlicht.

`<Directory /> ... </Directory>`

Ein `<Directory>`-Container enthält die Konfigurationsoptionen für ein einzelnes Verzeichnis. Der Container für das Wurzelverzeichnis bestimmt die Voreinstellung für *alle* Verzeichnisse und URLs. Es gibt einige Direktiven, die nur in Verzeichniscontainern stehen dürfen.

`Options None`

Die Direktive `Options` legt fest, welche »Dienstleistungen« in diesem Verzeichnis verfügbar sein sollen. Die Voreinstellung `None` für das Wurzelver-

zeichnis besagt, dass zunächst einmal sämtliche Optionen deaktiviert sind. Einzelheiten erfahren Sie im nächsten Kapitel.

`AllowOverride None`

Es besteht die Möglichkeit, Konfigurationsdirektiven in eine Datei innerhalb eines einzelnen Verzeichnisses oder sogar Unterverzeichnisses einer Website auszulagern. Diese Datei heißt standardmäßig `.htaccess` (kann durch die Direktive `AccessFileName` geändert werden). Am bekanntesten ist dieses Verfahren zur Erstellung passwortgeschützter Verzeichnisse; neben Authentifizierungsdirektiven lassen sich aber auch andere Einstellungen durch diese Datei lokal überschreiben.

`AllowOverride` legt fest, welche Direktiven überschrieben werden dürfen. Dazu werden diese nicht einzeln angegeben, sondern es gibt einige spezielle Bezeichnungen für Funktionsgruppen, die im nächsten Kapitel ausgeführt werden. Die Voreinstellung `None` bedeutet natürlich, dass zunächst einmal keinerlei Direktiven überschrieben werden dürfen. `.htaccess`-Dateien werden in diesem Fall sogar vollständig ignoriert.

`Order deny,allow`

Die Direktive `Order` aus dem Modul `mod_access` bestimmt, in welcher Reihenfolge die Zugriffseinstellungen `Allow` und `Deny` verarbeitet werden. `allow,deny` bedeutet, dass die Verbote die Erlaubnisse einschränken. Bei `deny,allow` ist es umgekehrt.

`Deny from all`

Die Direktive `Deny` stammt ebenfalls aus `mod_access`. Diese rigorose Einstellung besagt, dass auf das Wurzelverzeichnis zunächst einmal *niemand* zugreifen darf.

`<Directory /usr/local/apache2/htdocs> ... </Directory>`

Dieser Container enthält die Einstellungen für die freigegebene Website. Da das Wurzelverzeichnis sehr restriktiv abgesperrt wurde, müssen hier einige Einstellungen durch ausdrückliche Erlaubnisse überschrieben werden: `Options` und `AllowOverride` werden auf `All` gesetzt – unterhalb der `DocumentRoot` sind alle Verzeichnisoptionen verfügbar, und das Maximum an Direktiven darf in `.htaccess`-Dateien überschrieben werden. Order erhält den Wert `allow,deny` – die `Allow`-Einstellung soll Vorrang besitzen.

`DirectoryIndex index.html`

Diese durch das Modul `mod_dir` bereitgestellte Direktive definiert die Namen eines oder mehrerer Dokumente, die der Server ausliefert, wenn der

URL in einer Anfrage nur einen Verzeichnis-, aber keinen Dateinamen enthält. Falls mehrere Dateien angegeben werden, sucht Apache in der angegebenen Reihenfolge nach ihnen und liefert das erste Dokument aus, das er findet. `index.html` ist die Standardeinstellung.

`Allow from all`

Genau wie `Deny` ist auch `Allow` in `mod_access` definiert. Diese Zeile erlaubt allen Hosts aus dem gesamten Internet den Zugriff auf die Website.



## 5.3 Zusammenfassung

Das ausführbare Programm, das den Kern des Apache-Webservers bildet, lässt sich – typisch für Software aus der UNIX-Welt – mit zahlreichen Parametern und Optionen aufrufen. Sie können den Server damit nicht nur einfach starten, sondern über die bestehende Konfigurationsdatei hinaus anpassen oder auch wichtige Informationen über den Status des HTTP-Servers erhalten. Dies gilt sowohl für Apache auf der UNIX-Plattform als auch auf Windows-Systemen.

Daneben gibt es zahlreiche Hilfsmittel zur Steuerung des Webservers. Auf UNIX-Systemen ist er mit dem Shell-Skript `apachectl` ausgestattet, das nicht nur den angepassten Start, sondern auch das Beenden und den Neustart von Apache 2 ermöglicht. Unter Windows wurden einige dieser Optionen zum ausführbaren Programm (`apache.exe`) hinzugefügt.

Eine gewisse Herausforderung besteht in dem Problem, Apache beim Hochfahren des Systems automatisch zu starten. Auf einem UNIX-Rechner müssen Sie herausfinden, ob Ihr System System V Init oder die BSD-Boot-Methode verwendet, um die entsprechenden Startbefehle hinzuzufügen. Unter Windows empfiehlt sich zu diesem Zweck der Betrieb des Webservers als Dienst.

Nachdem Sie Ihren Server gestartet haben, sollten Sie mit einer Minimalkonfiguration wie der hier beschriebenen überprüfen, ob er auch ordnungsgemäß seine Arbeit erledigt, bevor Sie ihn praktisch im Internet oder Intranet einsetzen.

# Index

## B

## C

## D

## E

## F



## G



## H

## I

## M

## N



## O



## P

## Q



## R

## S

## T

## U



## V



## W