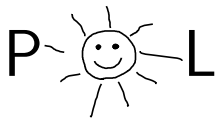
TCP/IP

Grundlagen · Applikationen · Sicherheit



Dr.-Ing. Peter Leibner

Tel.: +49 (0)3834 43 99 579
e-mail: pel@kedaj.org

Inhaltsverzeichnis

1 TCP/IP-Grundlagen

Den Anstoß zur Entwicklung der Kommunikationsprotokolle TCP/IP gab 1969 die staatliche amerikanische Agentur *Advanced Research Project Agency* (ARPA). Es sollte eine Technologie zur Übertragung von Datenpaketen über weitverteilte Netze implementiert werden. Entstanden ist zunächst ein Versuchsnetz namens ARPANET, das vier Standorte (die kalifornischen Universitäten von Los Angeles und Santa Barbara, das Stanford Research Institute und die Universität von Utah in Salt Lake City) miteinander verband. Den regelmäßigen Betrieb nahm ARPANET im Jahre 1971 auf.

Mit der wachsenden Bedeutung von militärischen Programmen wurde unter Führung der *Defense Communication Agency* (DCA) aus ARPA im Jahre 1972 DARPA (*Defense ...*). Im Jahre 1983 wurden die Protokolle TCP/IP als militärischer Standard der USA angenommen, zur selben Zeit entstand das *Internet*, in dem das ursprüngliche ARPANET ein Teilnetz bildete.

Das für die Verbreitung von TCP/IP wohl wichtigste Ereignis dürfte seine Integration in das Betriebssystem UNIX der *University of California at Berkeley* (UCB) gewesen sein. Diese erfolgte 1986 in die Version 4.3 BSD (*Berkeley Software Distribution*). Vor dieser Implementierung waren die Möglichkeiten der Kommunikation zwischen Rechnernetzen verschiedener Hersteller stark eingeschränkt, da die unterschiedlichen Hersteller alle ihre proprietären Technologien[1] verwendeten. Da viele Institutionen einen heterogenen Rechnerpark betrieben, stellte diese Inkompatibilität ein schwerwiegendes Problem dar.

Ein weiterer Grund für die erfolgreiche Verbreitung von TCP/IP ist seine offene Entwicklung. Alle wichtigen Standards werden in Form von RFC-Dokumenten (*Request For Comments*) veröffentlicht, so daß sie von jedermann implementiert werden können. Einige der erwähnten Rechnerhersteller nutzten diesen Vorteil zur Entwicklung von offenen Systemen, in denen in heterogenen Netzen die Rechner unterschiedlicher Hersteller problemlos zusammenarbeiten können.

1.1 Die Funktion von TCP/IP

Eine wichtige Aufgabe der Kommunikationsprotokolle ist die Abstrahierung des Übertragungsmediums. Diese erfolgt in Form eines Schichtenmodells (*Protocol Stack*). Da jede Schicht eine definierte Aufgabe der Übertragung übernimmt, kann sich ein Programmierer nun ganz auf die Programmierung seiner Applikation konzentrieren und muß sich nicht zusätzlich noch mit der Übertragungstechnik beschäftigen.

Die Schichten sind hierarchisch ihren Aufgaben entsprechend angeordnet (→ Abb.1.1).

[1] z.B. SNA (Systems Network Architecture) bei IBM, DECnet bei DEC, XNS (Xerox Network System) bei Xerox, SPX/IPX (Sequenced/Internet Packet Exchange) bei Novell, AppleTalk bei Apple oder VINES bei der Fa. Banyan

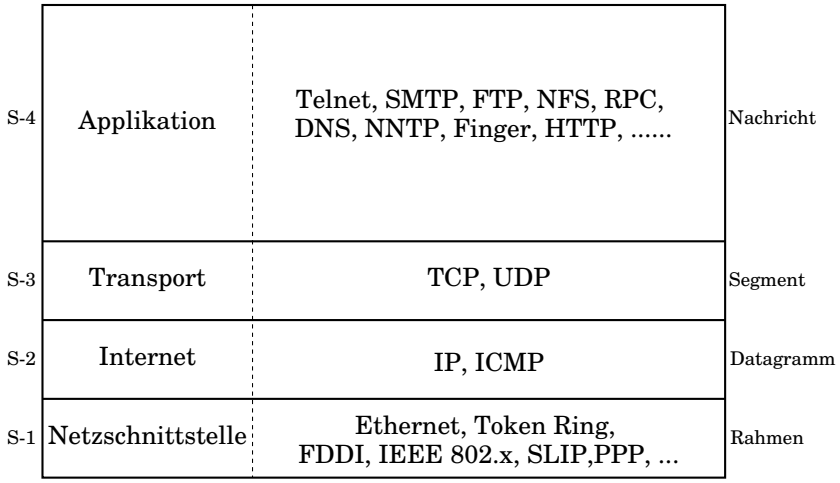


Abb. 1.1: TCP/IP Schichtenmodell

Auf beiden Seiten der Übertragungsstrecke kommunizieren jeweils die gleichen Schichten miteinander. Anweisungen bekommen sie von oben, Aufgaben erteilen sie an die jeweils niedrigere Schicht. Werden Nachrichten empfangen, so geschieht dies genau umgekehrt. Die Schicht bekommt Daten von der untergeordneten, entnimmt daraus die für sie bestimmten Informationen und gibt den Rest an die übergeordnete Schicht weiter.

Der Vorgang kann anhand der Kommunikation von zwei Direktoren zweier Firmen veranschaulicht werden. Bei der Zusammensetzung eines Briefes geht der Direktor davon aus, daß er seinem Gegenüber in der anderen Firma schreibt. Tatsächlich aber diktiert er den Brief einer Untergebenen, nämlich seiner Sekräterin. Diese weiß, daß den Brief in der anderen Firma ihre Kollegin erhält. Sie geht also davon aus, daß sie mit der gleichen Schicht auf der Gegenseite kommuniziert und fügt z.B. Angaben für die Korrespondenz an. Zum Transport gibt sie den Brief an die Post weiter. Diese leitet den Brief an den Adressaten weiter. Dabei kommt es nicht darauf an, auf welchem Wege oder in welcher Verpackung sie dies tut. In der anderen Firma empfängt die Sekräterin den Brief, entnimmt die für sie bestimmten Informationen und leitet den Brief an den Direktor weiter. Den jeweiligen Schichten ist es dabei völlig gleichgültig, was in den anderen Schichten mit dem Brief geschieht.

Unterschiedliche Netzarchitekturen unterscheiden sich in Anzahl und Aufgabe der jeweiligen Schichten. Als Referenzmodell für Netzarchitekturen dient die Anordnung nach OSI (*Open Systems Interconnection*). Dieses Modell liegt zur Standardisierung[2] vor und soll langfristig das TCP/IP-Schichtenmodell ersetzen. Bevor dies geschieht, ist das aktuelle Schichtenmodell das TCP/IP-Modell, das hier aus diesem Grunde auch besprochen wird.

S-1 Die Parameter und Funktionen der Netzschnittstelle (*Network Access Layer*) sind in der Regel fest in den Netzwerkadaptern konfiguriert. Erfolgt die Übertragung z.B. über das Ethernet, so wird auf dieser Schicht das Datagramm mit einem Kopf versehen und in einem Ethernet-Rahmen eingebettet über das Übertragungsmedium geschickt.

[2]Standard 7498 der *International Standardization Organization* ISO und des *International Electrotechnical Committee Standard* IECS aus dem Jahre 1984

Zur Verstärkung der Signale bei der Übertragung werden sogenannte *Repeater* eingesetzt. *Repeater*

Die Verbindung von Netzen erfolgt durch Brücken (*Bridge*). Die Brücke verfolgt die ankommenden Rahmen und trägt die in ihnen enthaltenen Adressen der Absender zusammen mit der Netzschnittstelle, von der sie den Rahmen erhalten hat, in eine Tabelle ein. Allmählich erhält sie so eine Zuordnung der einzelnen Rechner im Netz zu ihren Netzschnittstellen und kann ankommende Rahmen direkt auf die Netzschnittstelle weiterleiten, hinter der sich der Adressat befindet. Ist ein Adressat in der Tabelle noch nicht erhalten, so schickt sie den Rahmen an alle Netzschnittstellen, ausgenommen an die, von der sie den Rahmen erhalten hat. *Bridge*

S-2 Die Internet-Schicht (*Internet Layer*) ist für die Kommunikation von Stationen zuständig, die sich nicht im gleichen lokalen Netz befinden. Die Übertragung der Daten kann dabei über unterschiedliche Medien erfolgen, z.B. über das Ethernet auf eine serielle Verbindung und wieder zurück auf das Ethernet.

Die Verbindung von Netzen erfolgt in dieser Schicht durch *Router*. *Router*

Manchmal wird anstelle von *Router* auch der Begriff *Gateway* verwendet. In der Regel wird dieser Begriff dann verwendet, wenn damit die Verbindung von Netzen unterschiedlicher Protokollfamilien und damit eine notwendige Konvertierung der Protokolle im *Gateway*-Rechner verbunden ist (z.B. IP $\leftrightarrow$ AppleTalk). *Gateway*

Das Datagramm der Internet-Schicht wird vom Internet-Protokoll IP (*Internet Protocol*) durch das Hinzufügen von Informationen (Adressat, Absender, zuständiges Protokoll der darüberliegenden Schicht (*Assigned Number*, für TCP ist sie 6, für UDP 17), usw.) zum Kopf des Segments aus der darüberliegenden Schicht gebildet. Danach werden die einzelnen Datagramme unabhängig voneinander auf den Weg geschickt. Treffen sie unterwegs auf Übertragungsmedien, welche die ursprüngliche Länge eines Datagramms nicht übertragen können, so werden sie zerlegt. Eine Fragmentierung wird notwendig, wenn die zulässige *Maximum Transmission Unit* MTU (die Teil der IP-Konfiguration einer Netzschnittstelle ist) des folgenden Übertragungsmediums kleiner ist, als diejenige, aus dem das Datagramm kommt. Beim Übergang vom Ethernet (MTU=1500 Byte) auf eine serielle Verbindung (MTU=296 Byte), muß der *Router* das Datagramm folglich aufteilen. Die Teile werden über die Nummer des Datagramms und eine Fortsetzungsnummer für die Fragmente, identifiziert. Ihre Zusammensetzung erfolgt erst wieder in der Internet-Schicht des Empfängers. IP RFC 791 MTU

Die Datagramme einer Applikation können auf unterschiedlichen Wegen zu unterschiedlichen Zeiten beim Empfänger ankommen. IP versucht zwar, sie richtig zuzustellen, es garantiert aber nicht für das Ergebnis. Es ist daher notwendig, daß sich höhere Schichten gegen einen Verlust oder eine falsche Reihenfolge sichern. Bei IP handelt es sich um ein nichtverbundenes, unzuverlässiges Protokoll mit teilweiser Fehlererkennung ohne Fehlerkorrektur.

Eng verbunden mit dem Internet-Protokoll ist das *Internet Control Message Protocol* ICMP. Es dient für Mitteilungen über ICMP RFC 792

- die Unzugänglichkeit des Ziels (*Destination Unreachable*); Mitteilung an den Sender, daß das Netz, der Rechner, das Protokoll oder der Port nicht erreichbar

sind,

- die Herabsetzung der Datenübertragung (*Source Quench*); Mitteilung vom Empfänger oder *Router*, daß er nicht in der Lage ist, die ankommenden Daten zu verarbeiten und der Sender seine Sendegeschwindigkeit herabsetzen soll,
- den Wunsch auf Umlenkung des Datenstroms (*Redirect*); Mitteilung vom *Router*, daß ein Datagramm des Senders an einen anderen *Router* im selben Netz zur Weiterleitung geschickt werden soll (der Sender muß darauf seine *Routing*-Tabelle korrigieren),
- den Wunsch auf Echo und Antwort mit Echo (*Echo Request, Echo Reply*); dies erlaubt die Überprüfung der Zugänglichkeit von Stationen (z.B. mit dem Befehl **ping**),
- fehlerhafte Parameter (*Parameter Problem*); Mitteilung, daß ein Datagramm im IP-Kopf fehlerhafte Angaben hat und daher verworfen wurde.

Zur Übertragung der Mitteilungen nimmt ICMP die Dienste von IP in Anspruch. Es übernimmt daher von IP auch seine Eigenschaften. Es ist ein unverbundenes, unzuverlässiges Protokoll mit eigener Fehlerdetektion durch Bildung einer Kontrollsumme.

S-3 Die Transport-Schicht (*Transport Layer*) enthält zwei Protokolle, das *Transmission Control Protocol* TCP und das *User Datagram Protocol* UDP.

TCP
RFC 761

TCP übernimmt bei der Übertragung die Detektion und Korrektur von Fehlern, so daß eine zuverlässige Übertragung trotz Mängeln bei IP möglich wird. Bei TCP handelt es sich folglich um ein verbundenes, zuverlässiges Protokoll mit Fehlererkennung und Fehlerkorrektur.

Die hauptsächlichen Funktionen von TCP sind folgende:

- Verbindungsaufbau zwischen den kommunizierenden Rechnern vor Beginn der Datenübertragung und seine Unterbrechung nach seiner Beendigung.
- Durchschaltung der Transport- und Applikations-Schicht (*Application Layer*).

Port

Zur Identifikation der Programme der Applikations-Schicht, die miteinander kommunizieren wollen, dient ein Paar von *Ports*. Der Kopf des TCP-Segments enthält die Schlüssel des Sender-*Ports* und des Empfänger-*Ports* als seine ersten Elemente. Die *Ports* werden in Standard-*Ports* (*Well-Known Ports*) und ephemere *Ports* (*Ephemeral Port*) unterteilt. Die Schlüssel der Standard-*Ports* sind kleiner als 1024 und werden Diensten der Applikations-Schicht fest zugeordnet. Die ephemeren *Ports* werden nur vorübergehend (ephemer) einem Programm zugeordnet, wodurch es möglich wird, an ein und demselben Rechner gleichzeitig mehrere *Clients* der gleichen Applikation zu starten.
Beispiel. *Angenommen, es soll eine Telnet-Verbindung aufgebaut werden. Telnet hat den Standard-*Port 23*. Dazu startet die Station, welche die Verbindung aufbauen will (der* Client*), das Programm* **telnet***, dem es einen zufälligen* Port*, z.B.* 3068*, zuteilt. Der Empfänger-*Port *muß* 23 *sein, damit auf der Seite des* Servers *die Verbindung zum richtigen* Daemon *(Hintergrundprozeß, z.B.* **telnetd***) geknüpft werden kann. Die Antwort des* Servers *hat dann als Sender-*Port 23 *und als Empfänger-*Port 3068*. Die Verbindung zwischen*

den Stationen ist eindeutig. Es können daher mehrere Clients *über* Port 23 *mit dem* Server *Verbindung aufnehmen.*

- Bildung eines abstrakten Datenstromes (*Data Stream*). TCP unterteilt nicht wie IP die Daten in einzelne Datagramme, sondern betrachtet die Daten als sequentiellen Datenstrom. Solche Datenströme gibt es zwischen Sender und Empfänger immer zwei (hin und zurück).

- Bestätigung über Zustellung und Korrektheit der zugestellten Daten (*Positive Acknowledgement with Retransmission*). Um diese Aufgabe erfüllen zu können, numeriert TCP die einzelnen Bytes des Datenstroms durch. Der Anfangswert wird beim Verbindungsaufbau festgelegt. Jedes Segment enthält die Information über sein erstes Byte bezüglich der Reihenfolge der zu übertragenden Daten und seine Länge. Das Segment wird übertragen, gleichzeitig im Speicher des Senders aufbewahrt und ein Wecker gestellt.

  Der Empfänger schickt als Bestätigung den Wert das nächsten Bytes, das er in der zu übertragenden Reihenfolge des Datenstroms erwartet. Dieser Wert ist im Kopf des Segmentes enthalten, das der Empfänger an den Sender schickt. Im Idealfall ist dies ein Segment aus einem Datenstrom, der in Gegenrichtung fließt (das TCP-Protokoll ist vollduplexfähig). Wenn keine Daten an den Sender bereit stehen, wird ein leeres Segment mit der Bestätigung zurückgeschickt.

  Falls beim Sender nach Ablauf des Weckers keine Bestätigung eingegangen ist, so wird die Übertragung des letzten im Speicher aufbewahrten Segments wiederholt und der Wecker neu gestellt.

- Lenkung des Datenstromes (*Flow Control*). Dabei schickt der Empfänger dem Sender einen Parameter, genannt Fenster (*Sliding Window*), in dem er ihm mitteilt, wie viele Bytes in seinem Speicher noch Platz haben (obere Grenze). Dadurch erhält der Sender die Möglichkeit, weitere Segmente zu schicken, ohne auf die Bestätigung der vorhergehenden warten zu müssen.

  Als Fenster wird ein Ausschnitt aus dem Datenstrom bezeichnet, der mit dem ersten noch nicht bestätigten Byte anfängt und mit dem letzten Byte aufhört, das der Empfänger bereit ist zu empfangen. Durch ankommende Bestätigungen setzt der Sender die untere Grenze des Festers herauf, die obere Grenze darf nur der Empfänger erhöhen. Wird im Laufe der Übertragung die obere Grenze nicht erhöht, so kann es vorkommen, daß das Fenster geschlossen wird. Der Sender muß dann warten, bis er vom Empfänger ein Segment mit einem höheren Wert erhält.

UDP

Die Bestätigungsmechanismen von TCP erfordern einen nicht zu vernachlässigenden Fluß von Verwaltungsdaten. Für Applikationen, die ihre Daten nur in kleinen unabhängigen Paketen übertragen, ist es oft besser, UDP zu verwenden. UDP stellt eine nur geringe Erweiterung von IP dar. Es ist ein nichtverbundenes, unzuverlässiges Protokoll mit Fehlerdetektion.

Es ist in der Lage, Applikationen über *Ports* anzusprechen. Ähnlich wie bei TCP werden zur Übergabe der Daten an die Applikations-Schicht Sender-*Port* und Empfänger-*Port* verwendet (zu jeder IP-Adresse führt UDP 2^{16} *Ports* ein). Außer dieser

Angaben enthält der Kopf des UDP-Paketes nur noch die Gesamtlänge des Paketes und eine Kontrollsumme.

Der Begriff Paket stellt einen allgemeinen Terminus für eine Dateneinheit dar. Im Zusammenhang mit UDP ist damit ein Segment gemeint. Um die Verwirrung noch größer zu machen, wird ein UDP-Paket oft auch als Datagramm bezeichnet.

Das Protokoll UDP wird z.B. zur Implementierung von NFS (*Network File System*) verwendet. Mit Hilfe von NFS ist es möglich, auf Dateisysteme auf Rechnern im Netzwerk so zuzugreifen, als ob sie auf dem eigenen Rechner installiert wären.

NFS

S-4 Die Applikations-Schicht (*Application Layer*) stellt diejenige Schicht dar, in der die einzelnen Dienstprogramme realisiert sind. Jede Applikation hat sein eigenes Protokoll, mit dem es seine Aufgaben (z.B. elektronische Post, Datenübertragung, entfernte Anmeldung an einen Rechner im Netz, usw.) löst. Im Gegegensatz zu den niedrigeren Schichten, die universelle für alle gemeinsame Dienste anbieten, realisieren die Protokolle der Applikations-Schicht nur einen einzigen speziellen Dienst.

Die oben verwendeten Eigenschaften der Protokolle sind wie folgt zu deuten:

- Bei *verbundenen* Protokollen müssen die Partner vor der Übertragung der Daten eine Verbindung aufbauen, während bei *unverbundenen* Protokollen die Daten einem Vermittler übergeben werden, in der Hoffnung, daß dieser sie korrekt zustellt.

  Den Unterschied zwischen verbunden und unverbunden kann man beim Vergleich des Telefonierens (verbunden) mit der Briefpost (unverbunden) leicht erkennen.

- Bei *zuverlässigen* Protokollen wird die korrekte Zustellung von Daten bestätigt, bei *unzuverlässigen* nicht.

- Bei *Fehlererkennung* ist der Empfänger in der Lage festzustellen, ob ein bestimmter Fehler bei der Übertragung aufgetreten ist.

- Der einfachste Fall der *Fehlerkorrektur* ist die erneute Anforderung der fehlerhaften Daten. Es gibt jedoch auch intelligentere Methoden, wie z.B. fehlerkorrigierende Kode.

1.2 Struktur von IP-Adressen

Alle geläufigen Netzadapter haben eine bereits vom Hersteller fest vergebene Netzadresse. Die *Medium Access Control* MAC ist z.B. im Falle des Ethernets eine zwölfstellige Hexadezimalzahl. Die Zuteilung dieser Adressen ist weltweit koordiniert, so daß es keine zwei Adressen geben kann, die gleich sind.

MAC

Das Ethernet verwendet diese Adressen zur Adressierung im Rahmen eines lokalen Netzes. Da diesen Adressen jegliche Struktur fehlt, sind sie für eine weltweite Adressierung ungeeignet. Hat z.B. der lokale Netzadapter die Adresse `00:60:8c:51:c2:a9`, so kann ein zweiter, dessen Adresse um den Wert Eins höher liegt, am anderen Ende der Welt sein. TCP/IP verwendet daher nur in der Schicht der Netzschnittstelle im lokalen Netz Ethernetadressen, in der Internet-Schicht dagegen ein anderes, strukturierteres Adressierungsschema. Die Identifikation eines Netzadapters (ein Rechner kann mehrere haben)

erfolgt mit einer 32-Bit langen Zahl, die zur besseren Lesbarkeit Byteweise, mit einem Punkt zwischen den Bytes, als Dezimalzahl angegeben wird, z.B.,

$$\texttt{192.168.33.129} \Leftrightarrow (11000000101010000010000110000001)_2$$

Theoretisch ist es daher möglich, $2^{32} \approx 4.3 \cdot 10^9$ Rechner im *Internet* zu verbinden. Der Adreßraum hat jedoch eine feste Struktur[3], was das *Routing* erleichtert, die Anzahl der verfügbaren Adressen jedoch einschränkt.

1.2.1 IP-Netze

Ein Netz ist eine Gruppe miteinander verbundener Rechner, deren IP-Adressen eine bestimmten Anzahl von Anfangsbits gemeinsam haben. Diese Bits bilden die Netzadresse (*NetID*, → Abb.1.2). RFC 796 *NetID*

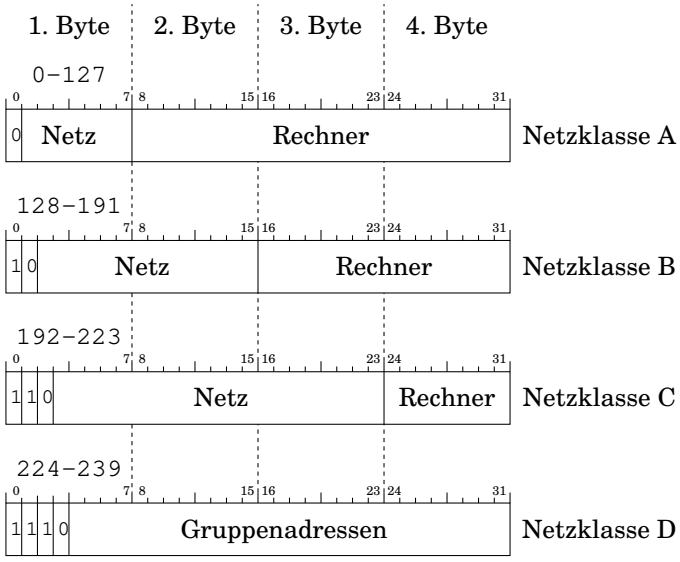


Abb. 1.2: Klassifizierung der IP-Adressen

Die restlichen Bits bilden die Adresse eines konkreten Rechners in diesem Netz (*HostID*, → Abb.1.2). *HostID*

Die Vergabe und Registrierung neuer IP-Adressen erfolgt in Deutschland durch das Deutsche Network Information Center (DeNIC[4]). DeNIC

Wie man aus Abbildung 1.2 entnehmen kann, unterscheiden sich die einzelnen Adressen in der Anzahl der adressierbaren Netze und Rechner. So können mit Adressen der Klasse A insgesamt 128 Netze mit jeweils fast 17 Millionen Rechnern adressiert werden. Diese Adressen wurden im ursprünglichen ARPANET zugeteilt, wo noch niemand ahnen konnte, welchen Zuwachs das *Internet* haben wird. Heute sind alle diese Adressen belegt.

[3] diese wird heute aus Mangel an Adressen nicht immer eingehalten, so daß es möglich ist, eine längere oder kürzere Netzadresse zu erhalten, die dann zusammen mit der Netzmaske zum *Routing* verwendet wird

[4] `http://www.nic.de/` am Rechenzentrum der Universität Karlsruhe (Tel. 0190 87 44 40, Fax. 04025 8194

Die Netzklasse B erlaubt die Adressierung von 16384 Netzen mit jeweils bis zu 65536 Rechnern. Auch diese Adressen sind zum größten Teil bereits belegt, so daß es nur bei Erfüllung besonderer Kriterien zur Zuteilung solch einer Adresse kommen kann.

Reelle Zuteilungsmöglichkeiten gibt es nur noch für die Klasse C. Diese erlaut die Adressierung von ca. 2 Millionen Netzen mit jeweils 256 Rechnern.

Einen Spezialfall stellt die Netzklasse D dar. Diese Klasse ist der Adressierung von Gruppen (*Multicast Address*) vorbehalten. Die Gruppenadressen sind applikationsspezifisch. Jede Adresse dieser Klasse ist für eine bestimmte Applikation, wie z.B. *Routing*-Protokolle, Netzspiele oder Videokonferenzen, reserviert.

Die eben gemachten Angaben zu den Adressenbereichen sind nicht ganz korrekt, da innerhalb des Adreßraums bestimmte Adressen für spezielle Zwecke reserviert und somit nicht zuteilungsfähig sind.

- Zu experimentellen Zwecken[5] dienen die Netzadressen `10.0.0.0`, `172.16.0.0`–`172.31.0.0` und `192.168.0.0`–`192.168.255.0`. Sie dürfen nicht zur Adressierung im *Internet*, können aber zur Adressierung innerhalb eines lokalen Netzes verwendet werden.
- Die Adresse `0.0.0.0` erscheint z.B. im Kopf eines Datagramms als Adresse des Absenders (sie darf nie als Zieladresse verwendet werden), das ein Rechner zur Feststellung seiner IP-Adresse in das Netz schickt (BOOTP, DHCP). Des weiteren kommt `0.0.0.0` besondere Bedeutung beim *Routing* zu.
- Ein ganzes Netz wird durch Adressen spezifiziert, die als Rechneradresse lauter Nullen enthalten, z.B. `150.111.0.0` (B-Netz).
- Die Adresse `255.255.255.255` wird zur Adressierung aller Rechner, die sich im gleichen Netz wie der Sender befinden (*Limited Broadcast*), verwendet. Das so adressierte Datagramm darf das Netz, z.B. über einen *Router*, nicht verlassen. Der Adresse `255.255.255.255` entspricht die Ethernet-Adresse `ff:ff:ff:ff:ff:ff`. Rahmen mit dieser Adresse müssen von allen Netzadaptern angenommen werden.
- Zur Adressierung aller Rechner in einem bestimmten Netz dienen Adressen, deren Netzadresse spezifiziert, deren Rechneradresse jedoch lauter Einsen enthält, z.B. `10.255.255.255`, `150.111.255.255` oder `200.111.222.255`.

  Die Adressierung aller Rechner in einem Netz, in dem sich der Sender befindet, bzw. die Adressierung aller Rechner in einem ausgewählten Netz (letzte zwei Punkte), stellen einen Spezialfall des *Multicast Addressing* (Netzklasse D) dar. Der Unterschied besteht darin, daß bei der Adressierung aller Rechner feststeht, um welche Rechner es sich handelt, beim *Multicasting* sich die IP-Adresse jedoch nicht auf einen speziellen Rechner bezieht (z.B. Ton- oder Bildübertragung eines Senders an mehrere Empfänger). Will z.B. eine bestimmte Applikation Datagramme einer konkreten Gruppe empfangen, so muß die Netzschnittstelle (Hardware

[5] Für das Demonstrationsnetz auf Seite 11 werden solche Adressen verwendet. In den darauf aufbauenden Beispielen wird jedoch angenommen, daß es sich bei den Adressen um ganz normale *Internet*-Adressen handelt, so daß alle Rechner auch über das *Internet* angesprochen werden können.

und Software) entsprechend konfiguriert sein. *Router*, die solche Datagramme weiterleiten müssen, erhalten Informationen über die richtige Netzschnittstelle mit Hilfe des *Internet Group Management Protocols* IGMP.

- Ein weiterer Spezialfall ist die Netzadresse `127.0.0.0`, bzw. die konkrete Adresse `127.0.0.1`. Sie wird zur Adressierung einer Schleife (*Loopback*) im lokalen TCP/IP-Betrieb des Rechners verwendet. So adressierte Datagramme gehen nicht über die Netzschnittstelle hinaus. Sie werden von lokalen Prozessen in der Internet-Schicht ausgetauscht, womit z.B. der lokale Betrieb von TCP/IP (z.B. mit **ping**) überprüft werden kann.

1.2.1.1 Subnetze

Zwischen den *Routern* werden Datagramme anhand der Netzadresse weitergeleitet, die Zustellung an einen bestimmten Rechner erfolgt erst am Zielort. Aus Mangel an genügend Adressen in den einzelnen Netzklassen, begann man die Netze weiter in Subnetze (*Subnet*) zu unterteilen bzw. Bereiche von Netzadressen zu Adreßbereichen zusammenzufassen (Supernetze, s.u.). Wird den so definierten Netzen eine Netzmaske hinzugefügt, so erscheinen sie für den *Router* (der die Maske auswerten kann) wieder wie ein zusammenhängendes Netz. RFC 1517-1520

Zur Bildung von Subnetzen wird am linken Rand der Rechneradresse eine bestimmte Anzahl von Bits für die *SubnetID* abgesondert. Die restlichen Bits bilden die *SubhostID* für die Adresse des Rechners innerhalb des Subnetzes. *SubnetID* *SubhostID*

Beispiel. *Für die Adresse* `172.24.233.12` *der Netzklasse B folgt z.B. die in Abb.1.3 angegebene Aufteilung.*

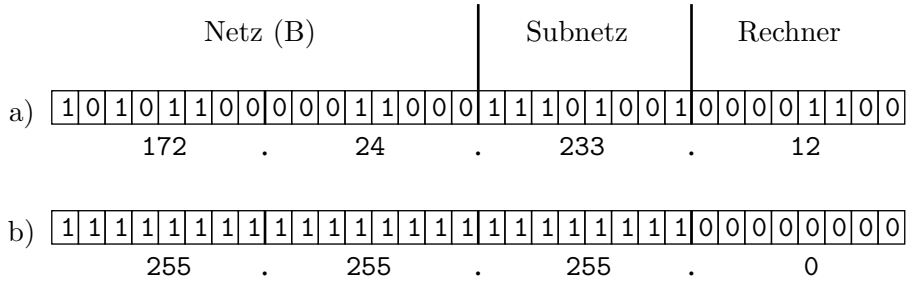


Abb. 1.3: a) Struktur der IP-Adresse, b) Subnetzmaske

Die Aufteilung in *SubnetID* und *SubhostID* muß nicht zwangsläufig an der Byte-Grenze erfolgen. Die Grenze zwischen Netz und Rechnern wird durch die Subnetzmaske (→ Bsp.1.2.1.1) bestimmt. Dabei handelt es sich um eine binäre Zahl, die im Bereich der *NetID* und *SubnetID* lauter Einsen, im Bereich der *SubhostID* lauter Nullen hat.

Wie bereits gesagt, adressieren lauter Nullen (z.B. `172.24.0.0`) im *HostID* ein ganzes Netz, lauter Einsen alle Rechner in einem bestimmten Netz (z.B. `172.24.255.255`). Für Subnetze gilt analog:

- Sind *NetID* und *SubnetID* spezifiziert und ist *SubhostID*=0, so ist das gegebene Subnetz gemeint.
  Beispiel. *Im Falle des C-Netzes mit der Netzadresse* 223.46.112.0 *und der Subnetzmaske*

$$255.255.255.\boxed{192} \Leftrightarrow (11111111111111111111111\boxed{11000000})_2,$$

  können grundsätzlich 4 Subnetze angegeben werden:

$$\begin{aligned}
223.46.112.\boxed{0} &\Leftrightarrow (11011111001011100111000\boxed{00000000})_2\\
223.46.112.\boxed{64} &\Leftrightarrow (11011111001011100111000\boxed{01000000})_2\\
223.46.112.\boxed{128} &\Leftrightarrow (11011111001011100111000\boxed{10000000})_2\\
223.46.112.\boxed{192} &\Leftrightarrow (11011111001011100111000\boxed{11000000})_2
\end{aligned}$$

- Besteht die *SubhostID* aus lauter Einsen und die *NetID* und *SubnetID* sind spezifiziert, so ist die Adressierung aller Rechner des gegebenen Subnetzes gemeint.
  Beispiel. *Um alle Rechner des Subnetzes* 223.46.112.192 *ansprechen zu können, muß daher die Adresse* 223.46.112.255 *verwendet werden.*

Wie die beiden letzten Beispiele zeigen, führt die Zulassung aller Subnetzadressen zu Problemen. Da die Adresse 223.46.112.0 für das ganze Netz steht, kann nicht die gleiche Adresse gleichzeitig für ein Subnetz verwendet werden. Da andererseits 223.46.112.255 alle Rechner im Netz 223.46.112.0 adressiert, können damit nicht nur die Rechner des Subnetzes 223.46.112.192 gemeint sein. In RFC 950 wird daher empfohlen, sowohl für *SubnetIDs* (*NetIDs*) als auch *SubhostIDs* (*HostIDs*) weder lauter Nullen noch lauter Einsen zu verwenden. Aus diesem Grund können im Beispiel 1.2.1.1 nur 254 statt 256 Subnetze mit jeweils 254 Rechnern statt 256 verwendet werden.

Die Einhaltung dieser Regel fällt insbesondere bei Netzen der Klasse C schwer. Es ist z.B. leicht einzusehen, daß eine Netzmaske

$$255.255.255.\boxed{128} \Leftrightarrow (11111111111111111111111\boxed{10000000})_2$$

unsinnig ist, da beide Subnetze, mit *SubnetID*=0 und *SubnetID*=128, eigentlich verboten sind. Es müssen folglich mindestens zwei Bit für das Subnetz spendiert werden, wobei man dann aber wegen der verbotenen 00 und 11 nur zwei Subnetze, nämlich 01 und 10 erhält. Die Hälfte des Adreßraums geht dabei verloren. Es wird daher empfohlen, wenigstens 3 Bit zur Adressierung von Subnetzen zu verwenden, da dabei abzüglich der beiden verbotenen Adressen immer noch sechs von acht Adressen übrig bleiben, was nur noch einem Ausfall eines Viertels der Adressen gleichkommt.

Beispiel. *In diesem Beispiel wird die Konfiguration eines kleinen Rechnernetzes eingeführt, das im folgenden als Referenz dienen wird. Es handelt sich dabei um ein kleines Firmennetz (→ Abb.1.4) mit der internen C-Netzadresse* 192.168.33.0.

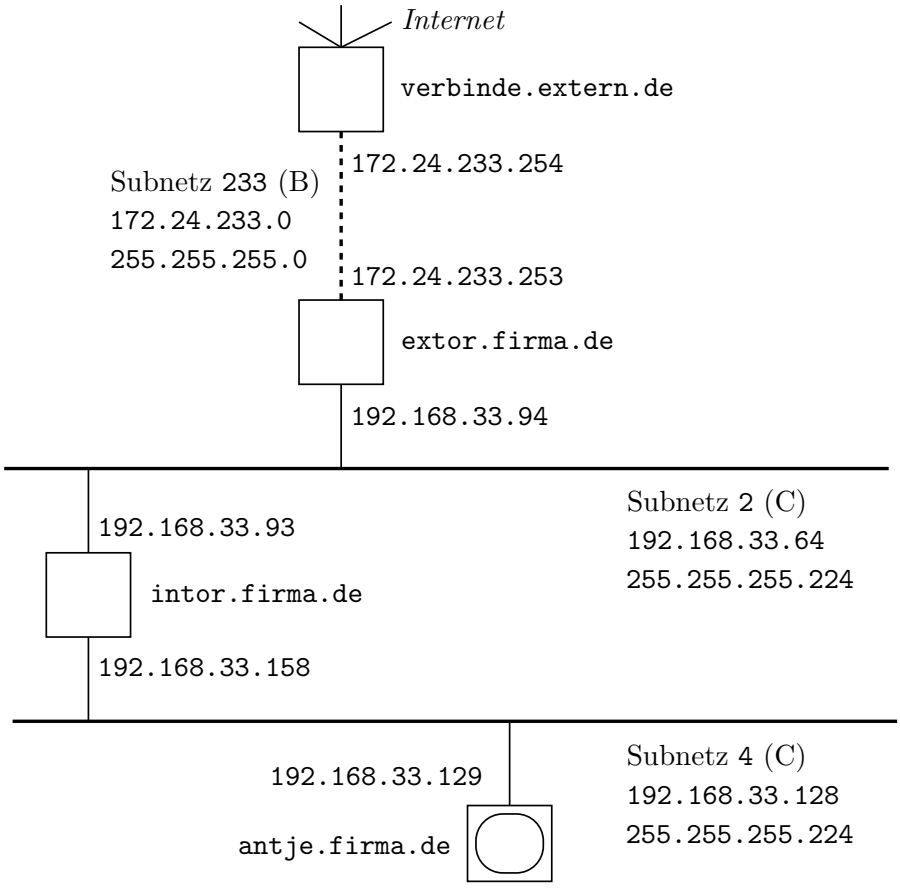


Abb. 1.4: Demonstrationsnetz

*Zur Zeit hat die Firma zwei Subnetze (*2 *und* 4*), mit der Subnetzmaske*

$$255.255.255.\boxed{224} \Leftrightarrow (11111111111111111111111\boxed{11100000})_2$$

Da drei Bit von der HostID *verwendet wurden, können bei Bedarf weitere Subnetze geschaffen werden. Der Rechner* `extor` *verbindet das Netz seriell mit dem Rechner* `verbinde`*, der an das* Internet *angeschlossen ist. Der Rechner* `intor` *verbindet die beiden internen Subnetze* 2 *und* 4*, der Rechner* `antje` *ist ein gewöhnlicher Benutzerrechner.*

(Die in Abb.1.4 verwendeten Domänennamen werden weiter unten behandelt.)

Die Verbindung in das Internet *geht über den* Router `verbinde` *eines* Internet-Providers*. Dieser spendiert hier ein ganzes Subnetz* (233) *seines B-Netzes* (172.24.0.0) *dafür. Dies ist ziemlich unreal, vielmehr würde es genügen, für den seriellen Anschluß zweier Rechner eine Subnetzmaske zu verwenden, die nur die letzten zwei Bit für die* SubhostID *beläßt. Die angegebene Adressierung wurde so gewählt, weil sie durch die offensichtliche Trennung der Netz- von der Rechneradresse übersichtlicher ist.*

Die Subnetzadressen sind

$$\texttt{192.168.33.}\boxed{\texttt{64}} \;\Leftrightarrow\; (11000000101010000010000\boxed{01000000})_2$$
$$\texttt{192.168.33.}\boxed{\texttt{128}} \;\Leftrightarrow\; (11000000101010000010000\boxed{10000000})_2,$$

für die Subnetzmaske folgt

$$\texttt{255.255.255.}\boxed{\texttt{224}} \Leftrightarrow (11111111111111111111111\boxed{11100000})_2.$$

Für die Subnetze wurden nur die ersten zwei Bit verwendet, so daß später die Subnetzmaske gegebenenfalls auf zwei Bit verkleinert werden kann. Dies wäre dann von Vorteil, wenn es sich herausstellen sollte, daß keine weiteren Netze eingerichtet werden müssen, die Anzahl der Rechner pro Netz aber über 30 ($2^5 - 2$) steigen sollte.

Bei den Adressen wurde die Konvention eingehalten, gewöhnliche Rechner von unten aufwärts, Router *und* Server *von oben abwärts zu zählen. In jedem Subnetz hat daher die nach außen führende Netzschnittstelle die* SubhostID=30 $\Leftrightarrow (00011110)_2$. *Konkret ergibt dies*

$$\texttt{192.168.33.}\boxed{\texttt{94}} \Leftrightarrow (11000000101010000010000\boxed{01011110})_2$$

für den Rechner `extor` *(Subnetz* 2*) und*

$$\texttt{192.168.33.}\boxed{\texttt{158}} \Leftrightarrow (11000000101010000010000\boxed{10011110})_2$$

für `intor` *(Subnetz* 4*).*

1.3 Routing

Die Entscheidung darüber, wohin ein Datagramm weitergeleitet werden soll, trifft der Rechner (oder *Router*) mit Hilfe eines *Routing*-Algorithmus anhand der Einträge in der *Routing*-Tabelle.

Routing-Tabelle

Beispiel. *Mit dem Kommando* **netstat** *und den Schaltern* **-rn** *kann die* Routing-*Tabelle ausgegeben werden. Der Schalter* **r** *bewirkt die Ausgabe der* `Kernel routing table`*, der Schalter* **n** *bewirkt, daß IP-Adressen nicht in Domänennamen überführt werden. Für den Rechner* `extor` *könnte die Tabelle wie folgt aussehen:*

netstat

```
extor:~$ netstat -rn
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use    Iface
192.168.33.64   0.0.0.0         255.255.255.224 U     0      0   1273   eth0
192.168.33.0    192.168.33.93   255.255.255.0   UG    0      0   3100   eth0
172.24.233.0    0.0.0.0         255.255.255.0   U     0      0   40918   sl0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0   12       lo
0.0.0.0         172.24.233.254  0.0.0.0         UG    1      0   127308  sl0
```

Die wichtigsten *Routing*-Informationen enthalten die ersten drei Spalten. Den Zeilenbeginn bildet jeweils die IP-Adresse des Zieles. Es kann sich um die Adresse eines Netzes, Subnetzes oder Rechners handeln. Handelt es sich um einen konkreten Rechner, so enthält `Flags` die Angabe `H` (*Host*). Weitere `Flags` sind `U` („up“, der Weg ist durchgeschaltet), `G` (der Weg ist indirekt, er geht über ein *Gateway*), `D` (der Weg wurde durch den Wunsch nach ICMP-*Redirect* in die Tabelle eingetragen) und `M` (der Weg wurde durch ICMP-*Redirect* geändert). Die Adresse `127.0.0.0` stellt die lokale Schleife, `0.0.0.0` den impliziten Weg (*Default Route*), dar. Dieser Eintrag wird immer dann verwendet, wenn es keinen spezifischeren gibt.

In der zweiten Spalte kommt `0.0.0.0` dreimal vor. Die IP-Adresse `0.0.0.0` gibt an, daß das Ziel im lokalen Netz bzw. in der lokalen Schleife liegt. Das so adressierte Datagramm kann daher direkt übergeben werden. Alle Datagramme, die nicht an das Subnetz `192.168.33.64` (Subnetzmaske `255.255.255.224`) adressiert sind, jedoch innerhalb des Netzes `192.168.33.0` zugestellt werden sollen, müssen über den Rechner `intor` (`192.168.33.93`) weitergeleitet werden, alle anderen (`0.0.0.0`) gehen über `verbinde` (`172.24.233.254`) aus dem lokalen Netz nach außen.

Routing-Maske

Die dritte Spalte bestimmt, wie allgemein die Zieladresse ist. Sie enthält die *Routing*-Maske, welche den Block IP-Adressen festlegt, der durch die gegebene Zeile vertreten wird. Die erste Zeile bezieht sich daher auf alle Adressen der Rechner, die im Subnetz `192.168.33.64` liegen, die zweite auf alle Rechner mit der Adresse `192.168.33.`x, wobei x jeden beliebigen Wert annehmen kann, die dritte auf alle Adressen mit dem Wert `172.24.233.`x, die vierte auf alle mit `127.`$x.x.x$ und die letzte auf den Rest ($x.x.x.x$). Für jede IP-Adresse eines Datagramms wird die Konjunktion mit den Werten in der Spalte `Genmask` gemacht. Der Vergleich des Ergebnisses mit den Zieladressen ergibt alle potentiellen Wege. Haben alle die gleiche Metrik, so wird der Weg gewählt, dessen Maske in der Spalte `Genmask` die meisten Einser hat (am spezifischsten ist).

Beispiel. *Der Rechner* `extor` *möchte ein Datagramm mit der Adresse* `192.168.33.129` *verschicken. Dazu führt er die Konjunktion der IP-Adresse des Datagramms mit den jeweiligen Werten in der Spalte* `Genmask` *durch.*

$$
\begin{array}{rrcll}
 & \texttt{192.168.33.129} & \Leftrightarrow & (11000000101010000010000110000001)_2 & \\
\wedge & \texttt{255.255.255.224} & \Leftrightarrow & (11111111111111111111111111100000)_2 & \\
\hline
 & \texttt{192.168.33.128} & \Leftrightarrow & (11000000101010000010000110000000)_2 & \text{(1.Zeile)} \\
 & & & & \\
 & \texttt{192.168.33.129} & \Leftrightarrow & (11000000101010000010000110000001)_2 & \\
\wedge & \texttt{255.255.255.0} & \Leftrightarrow & (11111111111111111111111100000000)_2 & \\
\hline
 & \texttt{192.168.33.0} & \Leftrightarrow & (11000000101010000010000100000000)_2 & \text{(2.Zeile)} \\
 & & & & \\
 & \texttt{192.168.33.129} & \Leftrightarrow & (11000000101010000010000110000001)_2 & \\
\wedge & \texttt{255.255.255.0} & \Leftrightarrow & (11111111111111111111111100000000)_2 & \\
\hline
 & \texttt{192.168.33.0} & \Leftrightarrow & (11000000101010000010000100000000)_2 & \text{(3.Zeile)}
\end{array}
$$

usw.

Aus der Konjunktion folgt als Zieladresse `192.168.33.0`*. Datagramme mit dieser Zieladresse müssen dem Rechner mit der IP-Adresse* `192.168.33.93` *übergeben werden.*

Für diesen folgt

$$\begin{array}{rrcll} & \texttt{192.168.33.93} & \Leftrightarrow & (11000000101010000010000101011101)_2 & \\ \wedge & \texttt{255.255.255.224} & \Leftrightarrow & (11111111111111111111111111100000)_2 & \\ \hline & \texttt{192.168.33.64} & \Leftrightarrow & (11000000101010000010000101000000)_2 & \text{(1.Zeile)} \end{array}$$

$$\begin{array}{rrcll} & \texttt{192.168.33.93} & \Leftrightarrow & (11000000101010000010000101011101)_2 & \\ \wedge & \texttt{255.255.255.0} & \Leftrightarrow & (11111111111111111111111100000000)_2 & \\ \hline & \texttt{192.168.33.0} & \Leftrightarrow & (11000000101010000010000100000000)_2 & \text{(2.Zeile)} \end{array}$$

usw.

Es kommen zwei Zieladressen in Frage. Da die `Genmask` *von* `192.168.33.64` *mehr Einsen enthält (spezifischer ist), wird dieser Weg gewählt und das Datagramm über die Netzschnittstelle* `eth0` *lokal* `(0.0.0.0)` *im Subnetz* `2` *an den Rechner* `192.168.33.93` *geschickt.*

Wie bereits oben erwähnt, können durch Netzmasken sowohl Teile der Rechner- als auch der Netzadressen ausgewählt werden. Während in dem einen Fall Subnetze entstehen, die z.B. ein LAN (*Local Area Network*) bilden können, ergeben sich im anderen Fall Supernetze (*Supernets*), die das *Routing* und die Administration von gemeinsamen IP-Adreßbereichen wesentlich vereinfachen. Diese Lösung wird klassenloses *Routing*, bzw. CIDR (*Classless Inter-Domain Routing*), genannt.

Benötigt z.B. eine Organisation mehr als eine Adresse für ein C-Netz, so kann ihr ein Block C-Netzadressen so zugeteilt werden, daß nach außen alle diese Netze wie ein Netz erscheinen. Dazu ist es notwendig, daß der Adreßblock zusammenhängend ist, d.h., über eine gemeinsame Anzahl von Anfangsbits adressiert werden kann und externe Systeme *Routing*-Protokolle verwenden, die in der Lage sind, neben der Zieladresse auch die Maske, die den Umfang des gültigen Adreßblocks für diese Zieladresse festlegt, auszuwerten.

Beispiel. *Ein zusammenhängender Block von 32 C-Netzadressen,*

$$\begin{array}{rcll} \texttt{194.123.}\boxed{\texttt{64}}\texttt{.0} & \Leftrightarrow & (1100001001111011\boxed{\texttt{01000000}}\texttt{.0})_2 & \textit{bis} \\ \texttt{194.123.}\boxed{\texttt{95}}\texttt{.0} & \Leftrightarrow & (1100001001111011\boxed{\texttt{01011111}}\texttt{.0})_2, & \end{array}$$

kann durch die Summenadresse `194.123.64.0` *und die* Routing-*Maske*

$$\texttt{255.255.}\boxed{\texttt{224}}\texttt{.0} \Leftrightarrow (1111111111111111\boxed{\texttt{11100000}}\texttt{.0})_2$$

eindeutig beschrieben werden, da alle Adressen in diesem Block die gleichen ersten 19 Bit von links haben.

Die Zusammenfassung von Adreßblöcken kann auf höheren Ebenen wiederholt werden, so daß bei günstiger Verteilung der IP-Adressen nach geographischen Regionen der Umfang der *Routing*-Tabellen der großen *Router* merklich reduziert werden kann. In RFC 1466 wird daher empfohlen, für Europa ausschließlich C-Netzadressen zu verwenden, die mit 194 oder 195 anfangen.

Die Metrik (*Metric*) in der fünften Spalte ist eine Art „Kostenfunktion“. Sie legt die Priorität der jeweiligen Wege fest. Für das *Routing* wird immer der Weg mit der niedrigsten Metrik gewählt. Die Auswahl der Wege mit gleicher Metrik erfolgt, wie oben beschrieben, nach dem Wert der `Genmask`.

`Ref` gibt die Anzahl der Referenzen auf diesen Weg an, `Use`, wie viele Datagramme über diesen Weg gesendet wurden und `Iface` die Netzschnittstelle, welcher das Datagramm übergeben werden soll. Die Kennzeichnung der Netzschnittstellen hängt von der UNIX-Version ab. So kann die lokale Schleife einmal als `lo`, ein andermal als `lo0` bezeichnet werden. Die Ethernet-Schnittstelle kann `eth`n, $n = 0, 1, \ldots$, oder `ne`n bzw. `ec`n heißen. Die für das gegebene UNIX-System gültige Bezeichnung erhält man mit dem Kommando **netstat -i**.

Einträge in *Routing*-Tabellen sind auf dreierlei Art und Weise möglich:

1. Eintrag eines statischen Weges (*Static Route*) mit Hilfe des Kommandos **route**. **route**
   Beispiel. *Für das Netz* `172.24.233.0` *lautet der Eintrag am Rechner* `extor` *z.B.*

   `extor:~#` **route add -net 172.24.233.0 dev sl0**

   Die Routing-*Maske bestimmt das Kommando in der Regel anhand der Netzadresse selbst. Ist dies nicht möglich, so muß sie hinzugefügt werden:*

   `extor:~#` **route add -net 192.168.33.64 netmask 255.255.255.224**

   In diesem Beispiel wurde wiederum die Netzschnittstelle weggelassen. Diese werden mit Hilfe des Kommandos **ifconfig** (Interface Configuration) *für alle Netzschnittstellen eines Rechners konfiguriert und stehen dann dem Kommando* **route** *zur Vervollständigung der Angaben zur Verfügung.* **ifconfig**

   `extor:~#` **ifconfig eth0 192.168.33.94 netmask 255.255.255.224 broadcast 192.168.33.95**

   Der Weg über ein Gateway *wird durch*

   `extor:~#` **route add -net 192.168.33.0 gw 192.168.33.93**

   und für den impliziten Weg durch

   `extor:~#` **route add default gw 172.24.233.254**

   eingetragen. Löschen kann man einen Eintrag aus der Routing-*Tabelle durch* **route del***, z.B.,*

   `extor:~#` **route del 192.168.33.0**

   Solch eine von Hand erstellte Konfiguration ist für die meisten Netze ausreichend. Wird das Netz um weitere *Router* erweitert, so wird die *Routing*-Tabelle mittels ICMP-*Redirect* automatisch aktualisiert.

2. Eintrag durch ICMP-*Redirect*.

3. Dynamisches *Routing* mittels *Routing*-Protokollen. Normale Rechner benötigen in der Regel keine *Routing*-Protokolle, sie kommen mit statischen *Routing*-Tabellen und ICMP-*Redirect* zurecht. *Routing*-Protokolle haben dagegen große Bedeutung für *Router*. Sie haben zwei Aufgaben:

- Austausch von *Routing*-Informationen mit anderen *Routern* und
- Aktualisierung der eigenen *Routing*-Tabelle.

1.3.1 Topologie des Internets

Im Falle des *Internets* entsteht das Problem, *Routing*-Tabellen gigantischen Ausmaßes verwalten zu müssen. Um die Größe auf einen verwaltbaren Rahmen zu reduzieren, wurden die oben erwähnten impliziten Wege eingeführt, die diejenigen Datagramme weiterleiten, für die es keinen expliziten Weg in der *Routing*-Tabelle gibt. Damit dabei keine Schleifen entstehen, muß eine Baumstruktur eingehalten werden. Es muß also eine Wurzel des Baums existieren, deren zentrale *Router* (*Core Gateways*) alle Wege in alle angeschlossenen Netze kennen. Solche *Router* gibt es im *Internet* eine beschränkte Anzahl.

Die Grundstruktur des *Internets* (→ Abb.1.5) besteht aus einem zentralen Rückgrat-Netz (*Core Backbone Network*) und einer Menge autonomer Systeme (*Autonomous Systems*).

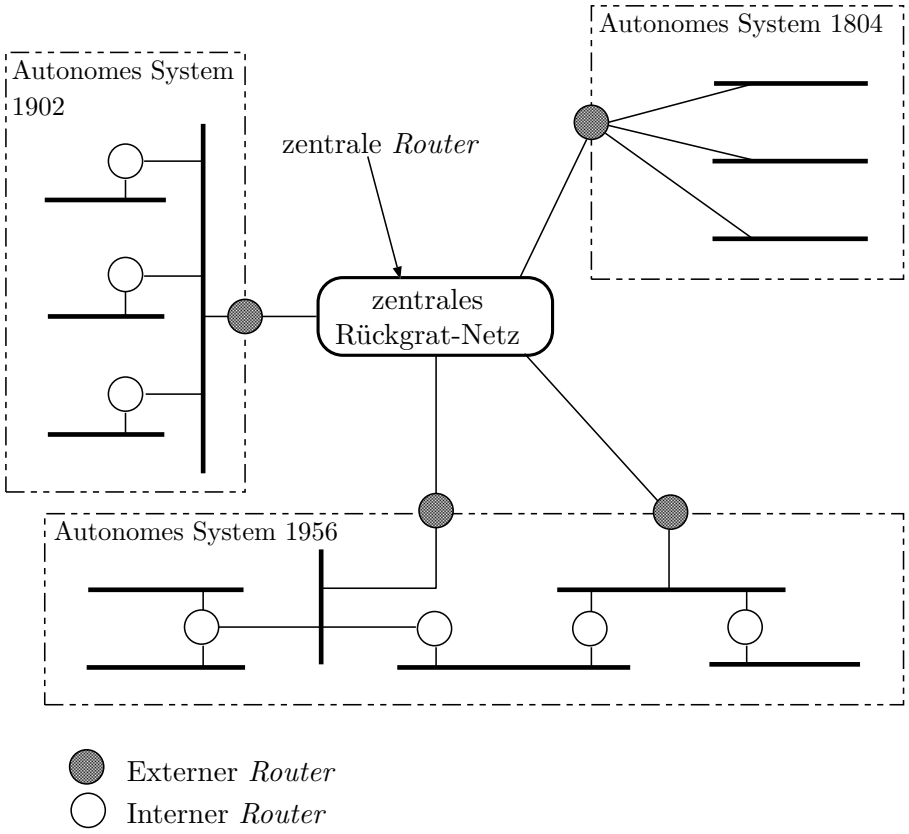


Abb. 1.5: Topologie des *Internets*

Ein autonomes System stellt eine Menge von Netzen und *Routern* dar, in denen ein konsistenter Mechanismus zum Sammeln von *Routing*-Informationen (ein spezielles internes *Routing*-Protokoll) und deren Verarbeitung und Weitergabe an andere autonome

Systeme, existiert. Autonome Systeme sind weltweit koordiniert, registriert und durch eine eindeutige Nummer identifiziert.

Das *Routing* erfolgt in zwei Ebenen. Zuerst innerhalb des autonomen Systems, dann zwischen den autonomen Systemen. Die *Routing*-Protokolle werden dementsprechend in interne (*Interior Routing Protocols*) und externe (*Exterior Routing Protocols*) unterteilt.

1.3.1.1 Interne Routing Protokolle

Das geläufigste interne Protokoll ist das *Routing Information Protocol* RIP. Der Austausch von *Routing*-Informationen erfolgt mittels UDP. Stationen, die am Austausch von *Routing*-Informationen beteiligt sind, werden in aktive und passive unterteilt. Aktive Stationen senden und empfangen Informationen, passive empfangen sie nur. Aktive sind daher in der Regel *Router*, passive normale Rechner (eine Netzschnittstelle). RIP

Router verschicken im aktiven Betrieb alle 30 Sekunden an alle angeschlossenen Netze mittels Adressierung aller Rechner[6] die ihnen bekannten *Routing*-Informationen (die Adressen zugänglicher IP-Netze und -Subnetze, sowie die ihnen zugeordnete Metrik). Die von RIP in die *Routing*-Tabelle eingetragenen Wege haben eine begrenzte Gültigkeit. Falls innerhalb einer festgelegten Zeitspanne (meistens 180 Sekunden) der Weg in den periodisch gesendeten Informationen nicht wieder vorkommt, so wird er aus der *Routing*-Tabelle wieder gelöscht. Dies verhindert sowohl den Überlauf der *Routing*-Tabellen als auch die Überlastung der Netze (es werden weniger Informationen verteilt).

Die Metrik gibt wieder, über wieviele *Router* ein Datagramm geschickt werden muß, um zur Zieladresse zu gelangen. Eine „unendliche“ Metrik stellt dabei für RIP ein unzugängliches Netz dar. Diese Grenze liegt bei bereits 16 *Routern*, wodurch sein Einsatz auf Netze, deren längste Verbindung bei 15 *Routern* liegt, beschränkt ist. Vorausgesetzt wird bei dieser Berechnung eine implizite Metrik von 1 pro *Router*. Es kann jedoch für einen bestimmten Weg auch ein höherer Wert auftreten. Dadurch kann z.B. eine schlechtere Übertragungsrate über eine serielle Verbindung bei der Bestimmung eines optimalen Übertragungsweges berücksichtigt werden. Dementsprechend reduziert sich die Anzahl der zulässigen *Router* und damit auch die Länge des maximal zulässigen Weges zwischen Sender und Empfänger in einem gegebenen Netz.

Neben seinen vielen Nachteilen[7], wie

- der Beschränkung des Weges auf maximal 15 *Router*,
- des *Routing* anhand der Metrik, welche die Durchlässigkeit eines Netzes nur grob wiederspiegelt,
- hohem Datenaufkommen bei der Übertragung von *Routing*-Informationen (ca. 20 Byte/Weg alle 30 Sekunden),
- der Unfähigkeit parallele Wege beim Datenstransport auszunutzen und damit die Netzbelastung aufzuteilen (es kennt immer nur den „besten“ Weg zum Ziel),

[6] beim neuen Protokoll RIP II (RFC 1388) erfolgt dies mit der reservierten Gruppenadresse `224.0.0.9`
[7] diese wurden zum größten Teil in RIP II beseitigt

- der beschränkten Verwendbarkeit von Subnetzmasken (da keine Informationen über Subnetzmasken übertragen werden, können nur Standardmasken verwendet werden),
- fehlender Autorisierungsmöglichkeiten für den Empfang von *Routing*-Informationen,

hat es zwei wesentliche Vorteile. RIP ist sehr einfach zu implementieren und wird von allen Herstellern unterstützt. Diese beiden Punkte machen RIP zum am weitesten unterstützten internen *Routing*-Protokoll in heutigen Systemen.

OSPF

Für kompliziertere Netze mit größerer Anzahl von *Routern* eignet sich besser als RIP das Protokoll *Open Shortest Path First* OSPF. Wie bereits der Name sagt, benutzt dieses Protokoll ebenso wie RIP den kürzesten Weg zwischen zwei Punkten. Gegenüber RIP, wo die Qualität der Verbindungen nur grob über die Metrik abgeschätzt wird, bewertet OSPF eine Verbindung anhand ihrer realen Durchgängigkeit.

Die Netze und *Router* eines autonomen Systems können in sogenannte OSPF-Gebiete (*OSPF Areas*) eingeteilt werden (→ Abb.1.6). Die einzelnen Gebiete enthalten mehrere Netze bzw. Subnetze. Jedes Gebiet ist mit dem OSPF-Rückgrat verbunden, was eine sternförmige Aufteilung ergibt. Gebiete, die nicht an das OSPF-Rückgrat angeschlossen werden können, nutzen eine virtuelle Verbindung zwischen einem *Router*, der am OSPF-Rückgrat angeschlossen ist und einem weiteren, der zu dem Gebiet gehört.

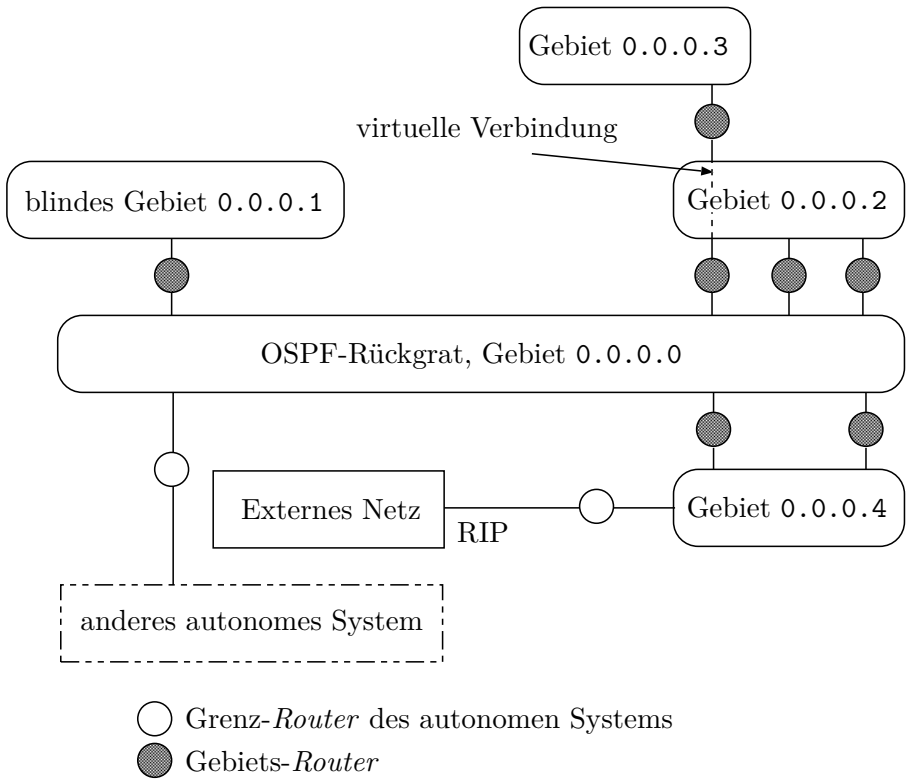


Abb. 1.6: Beispiel für ein autonomes System bei Verwendung von OSPF

Router, die verschiedene Gebiete verbinden, werden Gebiets-*Router* (*Area Border Router*, ABR), *Router*, die ein Gebiet zu einem anderen autonomen System oder externen Netz verbinden, Grenz-*Router* (*Autonomous System Boundary Router*, ASBR), genannt.

Blinde Gebiete (*Stub Areas*) kennen nach außen nur einen einzigen impliziten Weg. Der Gebiets-*Router* gibt nach innen nur den impliziten Weg weiter, der dann für alle Ziele außerhalb des blinden Gebietes verwendet wird. Solche Gebiete dürfen weder Grenz-*Router* haben noch dürfen durch sie virtuelle Verbindungen laufen.

Einzelne Gebiete werden durch 32-Bit lange Zahlen identifiziert. Ihre Schreibweise entspricht derjenigen von IP-Adressen, sie haben mit diesen aber nichts gemein.

Beim Start von OSPF untersucht der *Router* zunächst seine Umgebung. Dazu steht ihm das Subprotokoll OSPF-*Hello* zur Verfügung. Dieses hat die Aufgabe, benachbarte *Router* zu finden und Kontakt mit ihnen zu halten. Danach wird die eigene topologische Datenbasis (sie enthält Informationen über zugängliche *Router*, ihnen angeschlossene Netze, *Routing*-Masken, Metriken, usw.) mit den benachbarten *Routern* synchronisiert. Dies ist notwendig, da jeder *Router* seine *Routing*-Tabelle anhand dieser Datenbasis bildet. OSPF muß folglich dafür sorgen, daß allen *Routern* eines gegebenen Gebietes die gleichen Informationen zur Verfügung stehen. Nach erfolgter Synchronisierung werden zwischen den *Routern* nur noch wenig Informationen ausgetauscht. Ein Austausch geschieht erstens, wenn sich z.B. der Zustand einer Netzschnittstelle ändert oder die Topologie des Netzes erweitert wurde und zweitens in regelmäßigen Abständen von in der Regel 30 Minuten.

Zum Aufbau seiner Datenbasis ist OSPF auch in der Lage, Informationen von anderen *Routing*-Protokollen, wie RIP oder BGP (*Border Gateway Protocol*), auszuwerten. Diese werden dann im ganzen autonomen System verteilt.

RIP und OSPF unterscheiden sich wie folgt:

- OSPF eignet sich gegenüber RIP auch für große Netze mit nichttrivialer Struktur.
- Die durch OSPF verursachte Netzbelastung ist niedriger als bei RIP.
- OSPF ist in der Lage, mehrere Wege zwischen zwei Verbindungspunkten zu benutzen (*Load Splitting*) und anhand des geforderten Dienstes auszuwählen (*Type of Service Routing*).
- OSPF übernimmt Informationen über externe Wege von anderen *Routing*-Protokollen.
- Durch Einführung von OSPF-Gebieten wird hierarchisches *Routing* möglich. Da dabei die Topologie eines Gebietes den anderen verborgen bleibt, können Veränderungen in Gebieten unabhängig von den anderen durchgeführt werden. Alle Gebiete müssen mit dem OSPF-Rückgrat wenigstens eine Verbindung haben.
- Die Authorisierung einer empfangenen *Routing*-Information mittels Passwort wird unterstützt.
- Subnetzmasken können variabel sein (*Variable-Length Subnet Mask*).
- In Netzen, mit der Möglichkeit alle Rechner gleichzeitig zu adressieren (z.B. Ethernet), wird ein *Router* (*Designated Router*) ausgewählt, der dann über den Zustand der Leitungen alle anderen angeschlossenen *Router* verständigt. Dies reduziert die

Netzlast, da sonst diese Informationen alle *Router* miteinander austauschen müßten.

1.3.1.2 Externe Routing Protokolle

BGP Das zur Zeit wohl bekannteste externe *Routing*-Protokoll ist BGP. Seine Aufgabe besteht darin, *Routing*-Informationen zwischen den autonomen Systemen auszutauschen.

Interne *Routing*-Protokolle arbeiten mit Hilfe der Metrik, die für jedes autonome System konsistent ist. Bei externen Protokollen, die es mit einer Menge unterschiedlicher autonomer Systeme zu tun haben, kann dies nicht gewährleistet werden. BGP hält daher für jeden Weg auch die Reihenfolge der autonomen Systeme, über welche der Weg führt, in seiner Datenbasis bereit. Zum Austausch von Informationen kommt es zwischen sogenannten Nachbarn (*Neighbours*). Externe Nachbarn sind *Router*, die in unterschiedlichen autonomen Systemen liegen, die jedoch über ein gemeinsames Netz verbunden sind. Interne Nachbarn sind alle BGP-*Router* innerhalb eines autonomen Systems. Innerhalb eines autonomen Systems werden *Routing*-Informationen parallel mit den Informationen interner *Routing*-Protokolle, wie z.B. OSPF, verbreitet.

1.3.1.3 Lokales Routing im Ethernet

Die *Routing*-Algorithmen einer Sendestation oder eines *Routers* entscheiden darüber, was mit einem Datagramm zu geschehen hat, wie folgt:

1. Die Sendestation entscheidet anhand des Vergleichs der Ergebnisse der Konjunktion der eigenen IP-Adresse und derjenigen des zu versendenden Datagramms mit der Subnetzmaske, ob sich der Empfänger im lokalen Netz befindet.

2a. Befinden sich Sender und Empfänger im gleichen lokalen Netz, so wird das Datagramm direkt zugestellt.

2b. Befindet sich der Empfänger außerhalb des lokalen Netzes, so wird ein lokaler *Router* ausgesucht und diesem das Datagramm zur Weiterleitung übergeben.

Um Schritt 2a durchführen zu können, muß der Sender die Ethernetadresse des Empfängers im lokalen Netz kennen. Um diese festzustellen, sieht er in seiner Umsetzungstabelle, in der IP-Adressen Ethernetadressen zuordnet werden, nach. Zu diesem Zweck
ARP benutzt er das *Address Resolution Protocol* ARP, das diese Tabelle im Systemkern verwaltet.

Die Tabelle entsteht dynamisch. Falls der Sender feststellt, daß sich der Empfänger des Datagramms auf dem gleichen Segment des Ethernets befindet, schaut er in der ARP-Tabelle nach, ob dort bereits eine Ethernetadresse für die IP-Adresse des Empfängers eingetragen ist. Falls ja, so packt er das Datagramm in einen Ethernetrahmen und schickt es dem Adressaten. Falls nicht, schickt er an alle Rechner (mit der Adresse `ff.ff.ff.ff.ff.ff`) direkt (das Protokoll ist von IP unabhängig) einen speziellen Ethernetrahmen, der unter anderem folgende Angaben enthält:

- Es handelt sich um eine ARP-Anfrage[8].
- Die IP-Adresse des Adressaten.
- Die eigene Ethernetadresse.

Die betroffene Station ergänzt den empfangenen Rahmen um ihre Ethernetadresse und um die Mitteilung, daß es sich bei dem Rahmen um eine ARP-Antwort handelt. Diesen Rahmen schickt sie dann dem Sender, dessen Ethernetadresse sie ja aus der ARP-Anfrage kennt, zurück. Dieser trägt sich die Adresse in seine ARP-Tabelle ein und schickt dem Adressaten das IP-Datagramm.

Die Ausgabe der ARP-Tabelle erfolgt mit Hilfe des Kommandos **arp -a**. **arp**

Beispiel.

```
intor:~$ arp -a
Address                  HW type         HW address              Flags   Mask
192.168.33.93            10Mbps Ethernet 08:00:1E:04:C8:99       C       *
192.168.33.158           10Mbps Ethernet 08:00:2B:BC:09:E0       C       *
```

Bei `Flags` bedeutet der Buchstabe `C`, daß es sich um einen vollständigen gültigen Eintrag in der ARP-Tabelle handelt.

Es ist ferner möglich, eine Netzschnittstelle so zu konfigurieren, daß sie auch auf ARP-Anfragen reagiert, die eine fremde IP-Adresse enthalten. Eine solche Netzschnittstelle wird *Proxy*-ARP genannt. *Proxy*-ARP

Beispiel. *Angenommen, auf dem Rechner* `antje` *(→ S.11) läuft MS DOS und darunter eine Applikation, die mit einem verstümmelten IP-Protokoll arbeitet. Dieses kann keine Datagramme in ein anderes als das lokale Netz weiterleiten (fehlerhaftes* Routing*). Die Applikation kann daher Stationen, die sich im anderen Subnetz befinden, nicht ansprechen. Schickt z.B.* `antje` *eine ARP-Anfrage nach der IP-Adresse* `192.168.33.94`*, so bekommt sie keine Antwort. Abhilfe bringt die Eingabe von*

```
intor:~# arp -s 192.168.33.94 08:00:2B:BC:09:E0 pub
```

am Router `intor`*. Dies bewirkt, daß er nun auf ARP-Anfragen nach* `192.168.33.94` *mit seiner eigenen Ethernetadresse* `08:00:2B:BC:09:E0` *antworten wird.*

Proxy-ARP sollte nur mit Bedacht angewendet werden und sollte nicht als Ersatz für korrektes IP-*Routing* dienen. Es eignet sich vorwiegend für Stationen, die nur kurzzeitig via Modem an ein Netz angeschlossen werden (→ S.38).

1.4 Rechnerdomänen

Aus Sicht der Netzprogramme sind die numerischen IP-Adressen optimal, Benutzer ziehen jedoch in der Regel Rechnernamen vor, da sie sich leichter merken lassen. Die

Vergabe solch eines Names erfolgt mit dem Kommando **hostname** in einer der Initialisierungsdateien. **hostname**

Zur Überführung einer IP-Adresse auf einen Rechnernamen und umgekehrt, stehen unterschiedliche Mittel (*Name Services*) zur Verfügung. Am einfachsten geht es mit Hilfe einer Tabelle (**/etc/hosts**). Dieses Verfahren war anfangs die einzige Art der Zuordnung von IP-Adressen zu Rechnernamen. *InterNIC* hat zu diesem Zweck auf seinem FTP-*Server* die Datei **HOSTS.TXT** gehalten, in der alle registrierten Rechner eingetragen waren. Es war dann die Aufgabe jedes Systemverwalters, diese Datei auf allen seinen Rechnern regelmäßig zu aktualisieren. Angesichts der Größe des *Internets* ist dies jedoch heute nicht mehr realisierbar, so daß die Datei **hosts** in der Regel nur noch in Netzen verwendet wird, die nicht mit dem *Internet* verbunden sind.

NIS Im Rahmen des *Network Information Service* NIS[9] kann die Datei **hosts** für alle im lokalen Netz zusammengeschlossenen Rechner auf einem gemeinsamen NIS-*Server* gehalten werden. Dies hat den Vorteil, daß die Tabelle nur einmal aktualisiert werden muß, um dann von allen anderen Rechnern genutzt werden zu können.

DNS Für Rechner, die an das *Internet* angeschlossen werden sollen, kommt man praktisch nicht umhin, den *Domain Name Service* DNS zu verwenden. DNS funktioniert nach dem Prinzip *Client/Server*. Der *Server* stellt dabei eine Datenbasis mit Namen, IP-Adressen *Resolver* und weiteren Informationen zur Verfügung. Der *Client* (auch *Resolver* genannt) stellt eine Anfrage zunächst an den lokalen *Server*, von dem er dann entweder die gewünschte Antwort oder einen Verweis auf einen anderen DNS-*Server*, der näher an der Lösung des Problems liegt, erhält. Im letzteren Fall muß der *Client* dann seine Anfrage an diesen *Server* in der Regel wiederholen. Standardmäßig ist auf UNIX-Rechnern der BIND *Berkeley Internet Name Domain Server* BIND installiert. Der eigentliche *Daemon*, der den DNS-*Server*-Dienst realisiert, heißt **named**.

Unterschiedliche UNIX-Versionen setzen unterschiedliche Prioritäten bezüglich der Verwendung der eben angegebenen drei Methoden. Bei Linux besteht die Möglichkeit, die zu verwendende Reihenfolge für die Suche nach der IP-Adresse für den *Resolver* in der Datei **/etc/host.conf** vorzugeben.

Beispiel. *Der Eintrag*

```
order hosts,nis,bind
```

in der Datei **/etc/host.conf** *legt fest, daß zuerst die lokale Tabelle in* **/etc/hosts** *und falls der Name darin nicht enthalten ist, danach der NIS-*Server *und falls dies ebenfalls nicht zum Erfolg führt, zuletzt der DNS-*Server *zur Überführung der Namen auf IP-Adressen verwendet werden soll.*

[8] In Wirklichkeit ist das ARP-Protokoll etwas komplizierter, als hier angeführt. Es kann nämlich auch anderen Netzprotokollen als nur IP dienen. Da diese eine ganz andere Art der Adressierung haben können, wird bei einer ARP-Anfrage immer die IP-Adresse zusammen mit dem Typ des Protokolls angegeben.

[9] früher unter dem Begriff *Yellow Pages Service* bekannt (diese Bezeichnung kam in Konflikt mit der als „Yellow Pages“ für Branchen-Telefonbücher in Großbritannien eingetragenen Handelsmarke); Kommandos, die auf die gemeinsame Datenbasis zugreifen, fangen daher immer mit **yp**...an, z.B. **ypcat passwd** (zur Ausgabe der Datei **passwd** auf den Bildschirm)

1.4.1 Hierarchische Namensstruktur

Der DNS bildet eine konsistente baumförmige Struktur, deren Knoten Domänen genannt werden. Die Blätter dieses Baumes sind in der Regel konkrete Rechner. Die Wurzel stellt eine fiktive Domäne, die sogenannte Wurzeldomäne (*Root Domain*), dar. Ihr Name ist eine leere Zeichenkette. Jeder Knoten dieses Baumes wird durch eine Zeichenkette der Länge 0-63 (momentan nur 7-Bit ASCII) definiert. Die Namen können Groß- oder Kleinbuchstaben enthalten, bei Verwendung der Namen wird zwischen ihnen aber nicht unterschieden. Für benachbarte Knoten dürfen nicht dieselben Namen verwendet werden, ansonsten ist dies durchaus möglich.

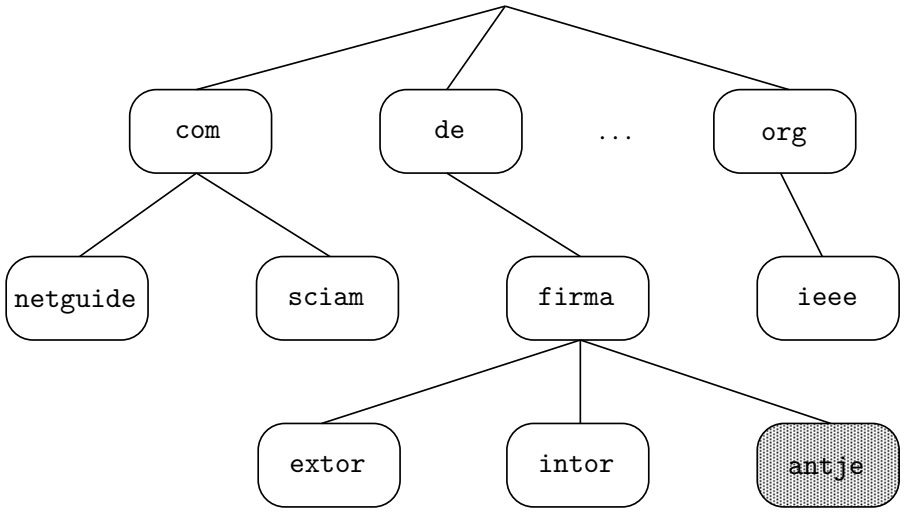


Abb. 1.7: Hierarchische Struktur der Domänennamen

Abbildung 1.7 stellt einen Ausschnitt aus dem System der Domänennamen dar. Die unmittelbar unterhalb der Wurzeldomäne liegenden Domänen werden Domänen der höchsten Hierarchiestufe (*Top-Level Domains*) genannt. Sie werden in thematische und nach Ländern geordnete, unterteilt (→ Tab.1.1).

`edu`	Bildungsanstalten
`com`	kommerzielle Unternehmen
`gov`	Regierungsstellen in den USA
`mil`	Militärische Einrichtungen in den USA
`net`	Verwaltungszentren des *Internets*
`int`	internationale Organisationen
`org`	restliche Organisationen
xx	Staatskode der einzelnen Staaten nach ISO 3166 (z.B. `de`, `uk`, `cz`,...)

Tab. 1.1: *Top-Level Domains*

Der Domänenname ist eine Zeichenkette, die eindeutig einen Knoten in der hierarchischen Struktur des DNS repräsentiert. Er setzt sich aus allen Bezeichnungen der Knoten

zusammen, die vom Ausgangsknoten bis zum Wurzelknoten durchlaufen werden müssen. Zwischen den Knotennamen wird jeweils ein Punkt geschrieben.

Beispiel. *Der schraffierte Knoten in Abb.1.7 hat folglich den Domänennamen*

`antje.firma.de.`,

wobei der abschließende Punkt die Bezeichnung der Top-Level Domain `de` *von der Wurzeldomäne (leere Zeichenkette) trennt*[10].

Domänennamen, die mit einem Punkt enden, werden als absolut bezeichnet und es wird angenommen, daß sie vollständig sind. Namen, die nicht mit einem Punkt enden, sind relativ und können von rechts, z.B. durch einen impliziten Domänennamen, vervollständigt werden.

Jeder Domäne kann eindeutig eine sogenannte Zone zugeordnet werden. Die Zone ist die Menge aller Domänen, die unterhalb der gegebenen Domäne liegen.

Beispiel. *In die Zone* `firma.de` *gehören folglich alle Namen, die mit* `firma.de` *enden.*

Für jede Zone muß es einen DNS-*Server* geben, der den hierarchischen Aufbau seiner Zone genau kennt. Dieser *Server* ist dann autoritativ für diese Zone. Diese Autorität erhält er von einem DNS-*Server* der darüberliegenden Zone.

Beispiel. *Angenommen, ein Benutzer am Rechner* `some.place.com` *möchte in Verbindung mit dem Rechner* `ftp.firma.de` *treten. Die Reihenfolge der Schritte könnte wie folgt aussehen:*

1. *Der DNS-*Client *des Rechners* `some.place.com` *muß die Adresse seines lokalen DNS-*Servers *kennen. An diesen wendet er sich dann mit der Bitte, „schicke mir die IP-Adresse der Domäne* `ftp.firma.de`*“.*
2. *Der lokale DNS-*Server *kennt die Adresse nicht. Er verweist daher den* Client *an einen der* Server *der Wurzeldomäne „.“, z.B. an* `a.root-servers.net`*. Die Adressen der DNS-*Server *der Wurzeldomäne muß jeder DNS-*Server *kennen (sie stehen in einer seiner Zonendateien)!*
3. *Der* Client *schickt die Anfrage nun an* `a.root-servers.net`*. Der* Server *kennt die IP-Adresse ebenfalls nicht, er kennt jedoch alle DNS-*Server*, die für die einzelnen Zonen der* Top-Level Domains *zuständig sind. Seine Antwort kann daher lauten: „weiß ich nicht, wende dich an* `ns.nic.de`*“.*
4. *Der* Client *wiederholt nun seine Anfrage an* `ns.nic.de`*. Auch er kann die Frage nicht beantworten, er kennt jedoch alle DNS-*Server *der Zone* `de`*. Er schickt daher die Antwort, „weiß ich nicht, frage bei* `extor.firma.de (192.168.33.94)`[11] *nach, der ist für die Zone* `firma.de` *zuständig“.*
5. *Diesmal adressiert der* Client *seine Anfrage an* `extor.firma.de`*. Da der Rechner* `extor.firma.de` *gleichzeitig DNS- und FTP-*Server *ist, erhält er die Antwort, „die IP-Adresse ist* `192.168.33.94`*“.*

[10] Der Punkt am Ende des Domänennamens wird in manchen Publikationen auch als Bezeichnung für die Wurzeldomäne interpretiert. Die im Beispiel erfolgte Erklärung hält sich sinngemäß an RFC 1035.

[11] siehe Fußnote auf Seite 8

Die im Beispiel angeführte Vorgehensweise wird nichtrekursiv genannt. Kennt ein *Server* die Antwort nicht, so verweist er den *Client* zum nächsten, näher am Ziel liegenden *Server*, an welchen der *Client* seine Anfrage dann wiederholt.

Neben der nichtrekursiven Vorgehensweise gibt es auch eine rekursive. In diesem Fall übernimmt die Suche nach der IP-Adresse der DNS-*Server* selbst. Die DNS-*Server* merken sich für eine begrenzte Zeit die ermittelten Adressen, so daß sie bei einer erneuten Anfrage eine nichtautoritative Antwort (autoritativ kann dies nur der zuständige Zonen-*Server*) geben können und somit nicht die ganze im Beispiel angegebene Prozedur wiederholt werden muß. Da dadurch die *Server* erheblich belastet werden können, unterstützen z.B. die *Server* der Wurzeldomäne diese Vorgehensweise nicht.

DNS hat nichts mit der Netzstruktur oder den IP-Adressen zu tun. Eine Firma, die sich an das *Internet* anschließt, erhält zwar eine IP-Adresse für ihr Netz sowie einen Domänennamen, was aber nicht heißt, daß z.B. ein Rechner in einem anderen als dem Firmennetz nicht einen Namen aus der Firmendomäne zugeteilt bekommen könnte. Die Beziehung zwischen IP-Adressen und Domänennamen ist auch in anderer Beziehung recht frei. So kann es für einen Namen mehrere IP-Adressen geben (`extor.firma.de` hat z.B. die IP-Adressen `192.168.33.94` und `172.24.233.253`) und eine Netzschnittstelle kann unter mehreren Namen angesprochen werden. Aufgrund dieser Möglichkeiten hat es sich eingebürgert, die Rechnernamen nach den angebotenen Diensten zu wählen. In der Gruppe der Namen muß es jedoch immer einen eindeutigen, sogenannten kanonischen Namen geben (z.B. `extor.firma.de`), für den es dann mehrere Alias-Namen, z.B. `ftp.firma.de` (FTP-*Server*), `www.firma.de` (WWW-*Server*) oder `ns.firma.de` („Nameserver“), geben kann.

In manchen Fällen ist es notwendig, für eine gegebene IP-Adresse den Namen des zugehörigen Rechners festzustellen. Zu diesem Zweck wurde die umgekehrte Domäne (*Reverse DNS*) `in-addr.arpa` geschaffen. Diese Domäne reiht die IP-Adressen der einzelnen Netze in eine Hierarchie entsprechend derjenigen des eigentlichen DNS ein. `in-addr.arpa`

Beispiel. *Die Zuordnung aller Adressen im Netz* `191.120.0.0` *zu ihren Domänennamen kann in der Zonendatei für die umgekehrte Domäne* `120.191.in-addr.arpa` *gefunden werden, für das Netz* `192.168.33` *ist dies die Datei* `33.168.192.in-addr.arpa`*. Der konkrete Name eines Rechners in diesem Netz, z.B. für die Adresse* `192.168.33.94`*, wird daher in der Zonendatei der Domäne* `33.168.192.in-addr.arpa` *gesucht.*

Es fällt auf, daß die Teile der dekadischen IP-Adressen in umgekehrter Reihenfolge geschrieben werden. Dies entspricht der Struktur des DNS, wo Subdomänen ebenfalls durch Erweiterung der übergeordneten Domänen von rechts nach links gebildet werden.

Manche Sicherheitsbarrieren erlauben den Netzzugang nur zu Stationen, die in der umgekehrten Domäne registriert sind. Dies gilt z.B. für manche anonyme FTP-*Server*, die eine Anmeldung nur dann gestatten, wenn für die IP-Adresse des *Clients* ein Name ermittelt werden kann.

Enthalten die Zonendateien für die umgekehrten Domänen die Namen der Rechner, die in diesen Zonen angesiedelt sind (so wie es sein sollte), so gibt das Kommando **traceroute** beim Verfolgen eines Datagramms vom Sender bis zum Empfänger die Rechnernamen an, über welche das Datagramm geleitet wurde und nicht ihre IP-Adressen. **traceroute**

Man könnte nun einwenden, daß ja gar keine feste Verbindung zwischen dem Sender und dem Empfänger besteht, daß folglich jedes Datagramm einen anderen Weg gehen wird[12]. Da sich die Einträge in den *Routing*-Tabellen jedoch eine gewisse Zeit halten, ist es sehr wahrscheinlich, daß alle Datagramme einer Sitzung über den gleichen Weg geleitet werden.

Der Vorteil von **traceroute** gegenüber **ping** liegt darin, daß es alle Rechner ausgibt, über welche das Datagramm gelaufen ist. Bei einem Fehler, der von **ping** lediglich durch `Host is unreachable` kommentiert wird, kann bei **traceroute** immerhin festgestellt werden, bis wohin das Datagramm fehlerfrei geleitet werden konnte.

Beispiel.

```
antje:~$ traceroute 172.24.233.254
traceroute to verbinde.extern.de (172.24.233.254), 30 hops max, 40 byte packets
 1  intor.firma.de (192.168.33.158)  2.542 ms  2.371 ms  1.718 ms
 2  extor.firma.de (192.168.33.94)  3.124 ms  3.214 ms  3.402 ms
 3  verbinde.extern.de (172.24.233.254)  56.521 ms  57.057 ms  63.561 ms
```

Jeder Station auf dem Weg schickt **traceroute** *drei Datagramme und gibt aus, wie lange es gedauert hat, bis es eine Antwort erhalten hat. Die Antworten von* **intor** *und* **extor** *sind relativ schnell, da die Übertragung über das Ethernet erfolgte, die Antwort von* `verbinde`*, der an der seriellen Leitung hängt, dagegen relativ langsam.*

1.4.2 Konfiguration des Resolvers

Für die korrekte Funktion im DNS-System benötigt der *Client* bzw. *Resolver* lediglich zwei Angaben: seinen eigenen Domänennamen und die IP-Adresse seines DNS-*Servers*. Diese Angaben werden in die Datei **/etc/resolv.conf** geschrieben. Die Implementierung des *Resolvers* erfolgt unter UNIX durch Systembefehle (*System Calls*), wie z.B. **gethostbyname** oder **gethostbyaddr**. Ein Benutzerprogramm, das diese Funktionen aufruft, braucht sich folglich nicht darum zu kümmern, ob der gegebene Rechner einen DNS-*Server* (BIND), NIS oder die Tabelle in **/etc/hosts** verwendet.

Beispiel. *Der Rechner* **antje** *könnte folglich als reiner Konsument der DNS-Informationen in seiner Datei* **/etc/resolv.conf** *folgende Angaben haben:*

```
# Konfiguration des DNS-Resolvers
domain firma.de
nameserver 192.168.33.94
nameserver 172.24.138.1
```

*Dabei wird vorausgesetzt, daß der DNS-*Server *für die Zone* `firma.de` *auf dem Rechner* `extor` *installiert ist und daß der* Internet-Provider *als Alternative seinen eigenen* Server *mit der Adresse* `172.24.138.1` *zur Verfügung stellt.*

Die in der Datei **/etc/resolv.conf** möglichen Angaben hängen von der jeweiligen UNIX-Implementation ab. Unter Linux sind unter anderen folgende Angaben erlaubt:

[12] Im Gegensatz zu einer festen Verbindung zweier kommunizierender Partner im Telefonnetz (*Circuit Switching*), besteht im *Internet* zwischen Sender und Empfänger nur eine „feste“ logische Verbindung (*Packet Switching*).

`domain` Durch `domain` wird die Domäne definiert, in der sich der lokale Rechner befindet. Falls die Angabe fehlt, versucht der *Resolver* die Angabe durch den Aufruf des Systembefehls **getdomainname** zu ermitteln.

`nameserver` Solche Zeilen kann es mehrere geben. Die angegebenen „Nameserver“ werden in der Reihenfolge ihres Auftretens in der Datei **/etc/resolv.conf** kontaktiert. Fehlt die Angabe `nameserver`, so versucht der *Resolver* einen DNS-*Server* zu kontaktieren, der auf ihm selbst installiert ist.

`search` Falls ein Programm versucht, einen relativen Namen zu ermitteln (ohne Punkt am Ende) und der DNS-*Server* kann diesen Namen nicht finden, so versucht der *Resolver* nacheinander die bei `search` angegebenen Domänen,

z.B., `search` *`domain_1 domain_2 ...`*,

an den Namen anzuhängen und die Anfrage an den DNS-*Server* zu wiederholen. Fehlt die Zeile, so wird die Angabe aus `domain` verwendet. Dank dieser automatischen Vervollständing des Domänennamens reicht es z.B. **ping intor** anstatt des vollständigen Namens **intor.firma.de** anzugeben.

1.4.3 Konfiguration des DNS-Servers

Der „Nameserver“ schöpft seine Informationen aus zwei Quellen. Zum einen sind dies die Daten, die auf ihm selbst abgelegt sind, so er für eine bestimmte Zone autoritativ ist, zum zweiten sind es Daten, die er bei der Erledigung einer Anfrage ermittelt hat (wenn er rekursiv arbeitet) und dann für einige Zeit zwischenspeichert (*cached*).

Für DNS-*Server* werden folgende Betriebsweisen unterschieden:

passiv Ein passiver (*caching-only*) DNS-*Server* hat keine autoritativen Zoneninformationen. Alle Informationen erhält er von anderen DNS-*Servern*, die ermittelten Daten merkt er sich jedoch für einige Zeit. Erfolgt die gleiche Anfrage später nochmal, so leitet er sie nicht weiter, sondern beantwortet sie selbst. Diese Antwort ist jedoch ohne Garantie. Alle DNS-*Server* sind zumindest passive „Nameserver“, auch wenn sie für andere Zonen eine der nachfolgenden Aufgaben erfüllen.

primär Jede Zone muß einen primären DNS-*Server* haben, der dann für diese Zone autoritativ ist. Er bestimmt durch seine Dateien den Inhalt der Zone. Die Dateien werden vom Systemverwalter für diese Zone zusammengestellt.

sekundär Der sekundäre DNS-*Server* dient als Reserve für den Fall, daß der primäre aus irgendeinem Grund ausfallen sollte. Auch er ist für eine bestimmte Zone autoritativ, seine sämtlichen Informationen muß er sich jedoch vom zuständigen primären *Server* holen. Dies geschieht immer dann, wenn er bei seinen regelmäßigen Kontrollen des primären *Servers* eine Veränderung in seinen Daten feststellt.

Für jede Zone muß ein sekundärer *Server* installiert sein.

Ein Exemplar des *Daemons* **named** kann gleichzeitig mehrere Zonen in unterschiedlichen Regimen (er kann z.B. für eine Zone Primärserver, für eine andere Sekundärserver, sein) bedienen.

Die Initialisierungsdatei für **named** heißt **/etc/named.boot**. Sie enthält hauptsächlich Verweise auf die Zonendateien bzw. auf andere DNS-*Server*.

Passiver Server

In diesem Fall konsumiert der Rechner die DNS-Angaben nur, es läuft aber auf ihm ein DNS-*Server*, so daß er über einen Ausgleichsspeicher verfügt, wodurch wiederholte Anfragen schneller beantwortet werden können.

Beispiel. *Die Initialisierungsdatei für einen passiven* Server *könnte wie folgt aussehen:*

```
directory /etc/namedb
;Typ       Domäne                 Quelle
;
primary    0.0.127.in-addr.arpa   db.127.0.0
cache      .                      db.cache
```

`directory` *Die Anweisung* `directory` *gibt an, wo die Zonendateien gesucht werden sollen.*

Der passive Server *wird als autoritativ für die Domäne* `0.0.127.in-addr.arpa` *definiert. Im Netz* `127.0.0.0` *ist unter UNIX die Adresse für die lokale Schleife (→ S.9) enthalten. Durch die obige Angabe wird es den* Clients*, die zur Überführung von Namen auf IP-Adressen ausschließlich auf den DNS-*Server *vertrauen, ermöglicht, der Adresse* `127.0.0.1` *den kanonischen Namen* `localhost` *zuzuordnen.*

`primary` *Das verwendete Schlüsselwort* `primary` *hat zwei Parameter, der erste ist der Name der Domäne, der zweite der Name der zugehörigen Zonendatei.*

`cache` *Das Schlüsselwort* `cache` *hat ebenfalls zwei Parameter. Der erste ist der Name der Wurzeldomäne (repräsentiert durch „.“) und der zweite derjenige der Zonendatei, welche alle zur Initialisierung des Ausgleichsspeichers notwendigen Informationen über die DNS-*Server *der Wurzeldomäne enthält. Da diese* Server *für das gesamte* Internet *dieselben sind, sollte diese Datei für alle DNS-*Server *ebenfalls die gleiche sein. Der Systemverwalter ist daher angehalten, die zentrale Datei*[13] *regelmäßig zu überprüfen und seine lokale Datei* **db.cache** *gegebenenfalls anzupassen.*

Mitte 1997 sollte die Datei **db.cache** *wie folgt aussehen:*

```
;/etc/namedb/db.cache
;Datei zur Initialisierung des Ausgleichsspeichers des DNS-Servers
;
;Name                    ttl      Klasse Typ   Daten
;
.                        3600000  IN      NS    A.ROOT-SERVERS.NET.
                         3600000  IN      NS    B.ROOT-SERVERS.NET.
                         3600000  IN      NS    C.ROOT-SERVERS.NET.
                         3600000  IN      NS    D.ROOT-SERVERS.NET.
                         3600000  IN      NS    E.ROOT-SERVERS.NET.
                         3600000  IN      NS    F.ROOT-SERVERS.NET.
                         3600000  IN      NS    G.ROOT-SERVERS.NET.
                         3600000  IN      NS    H.ROOT-SERVERS.NET.
```

[13] `ftp://ftp.rs.internic.net/domain/named.root`


```
                        3600000  IN      NS    I.ROOT-SERVERS.NET.
                        3600000  IN      NS    J.ROOT-SERVERS.NET.
                        3600000  IN      NS    K.ROOT-SERVERS.NET.
                        3600000  IN      NS    L.ROOT-SERVERS.NET.
                        3600000  IN      NS    M.ROOT-SERVERS.NET.
;
A.ROOT-SERVERS.NET.     3600000          A     198.41.0.4
B.ROOT-SERVERS.NET.     3600000          A     128.9.0.107
C.ROOT-SERVERS.NET.     3600000          A     192.33.4.12
D.ROOT-SERVERS.NET.     3600000          A     128.8.10.90
E.ROOT-SERVERS.NET.     3600000          A     192.203.230.10
F.ROOT-SERVERS.NET.     3600000          A     192.5.5.241
G.ROOT-SERVERS.NET.     3600000          A     192.112.36.4
H.ROOT-SERVERS.NET.     3600000          A     128.63.2.53
I.ROOT-SERVERS.NET.     3600000          A     192.36.148.17
J.ROOT-SERVERS.NET.     3600000          A     198.41.0.10
K.ROOT-SERVERS.NET.     3600000          A     193.0.14.129
L.ROOT-SERVERS.NET.     3600000          A     198.32.64.12
M.ROOT-SERVERS.NET.     3600000          A     202.12.27.33
```

*Für die DNS-*Server *der Wurzeldomäne wurde eine eigene Domäne* `root-servers.net` *angelegt. Die* Server *sind nicht nur in den USA stationiert, sie können sich irgendwo auf der Welt befinden (*`L.ROOT-SERVERS.NET.` *ist z.B. in Japan lokalisiert).*

Das Semikolon (`;`*) leitet einen Kommentar ein, der Dateiname und der Name für das Verzeichnis, in das alle Zonendateien abgelegt werden, können frei gewählt werden.*

Ein DNS-*Server* kann auch auf einem Rechner betrieben werden, der keinen direkten Zugriff auf das *Internet* hat. In solch einem Fall wird eine Nachfrage mittels des Eintrags `forwarders` in der Datei **/etc/named.boot** an die dort aufgeführten *Server* weitergeleitet. Die *Server* werden der Reihe nach so lange kontaktiert, bis einer in der Lage ist, die Frage zu beantworten. Der entfernte (z.B. auf einer „Firewall“ sitzende) *Server* versucht die Antwort selbst festzustellen (er arbeitet rekursiv) und gibt sie dann gegebenenfalls an den fragenden weiter. Es ist selbstverständlich notwendig, daß die Systemadministratoren der bei `forwarders` angeführten *Server* mit der Benutzung ihrer Rechner einverstanden sind.

`forwarders`

Im Zusammenhang mit `forwarders` sei noch `slave` erwähnt. Dieses Schlüsselwort hat keine Parameter. Es definiert den *Server* lediglich als untergeordnet (*Slave Server*). Seine Verwendung macht nur zusammen mit `forwarders` Sinn. Der untergeordnete *Server* versucht dann nicht seine Anfragen an autoritative *Server* für eine bestimmte Zone zu schicken, sondern wendet sich immer nur an die bei `forwarders` eingetragenen.

`slave`

Für den DNS-*Server* in seiner Rolle als *Client* kann in der Datei **/etc/resolv.conf** die lokale Adresse,

```
nameserver 127.0.0.1
```

eingetragen werden[14].

[14] siehe Bemerkung zu `nameserver` auf Seite 27

Primärer Server

Beispiel. *Für die Domäne* `firma.de` *soll* `extor` *als primärer* Server *eingerichtet werden. Seine Initialisierungsdatei kann dann wie folgt aussehen:*

```
directory /etc/namedb
;Typ       Domäne                    Quelle
;
primary    firma.de                  db.firma.de
primary    33.168.192.in-addr.arpa   db.192.168.33
primary    0.0.127.in-addr.arpa      db.127.0.0
cache      .                         db.cache
```

Die entscheidenden Informationen zur Konfiguration sind in den Zonendateien, die den Aufbau der jeweiligen Zonen wiedergeben, enthalten. Die Informationen zu den jeweiligen Zonen werden in einem festen Format (*Resource Record*) abgelegt. Es hat den folgenden allgemeinen Aufbau:

[*Domäne*] [*ttl*] [*Klasse*] *Typ Daten*

Domäne Gibt den Namen der Domäne an, auf die sich der *Record* bezieht. Der Name kann entfallen, sofern er der gleiche wie beim vorhergehenden *Record* ist. Der Name kann absolut (mit abschließendem Punkt) oder relativ angegeben werden. Dem relativen Namen wird der implizite Domänenname, auf den sich die Zonendatei bezieht (→ **named.boot**), angefügt. Die implizite Domäne wird in der Zonendatei durch `@` repräsentiert.

ttl Bestimmt die Gültigkeitsdauer (*time-to-live*) des *Records* in Sekunden (max. achtstellige Dezimalzahl). Nach Ablauf dieser Zeit wird der Eintrag aus dem Ausgleichsspeicher entfernt. Dadurch wird vermieden, daß in den DNS-*Servern* veraltete Informationen gespeichert bleiben, die dann eventuell Inkonsistenzen im DNS herbeiführen könnten. Fehlt dieser Eintrag, so wird *minimum_ttl* aus dem vorhergehenden `SOA`-*Record* übernommen.

Klasse Legt die Adressierungsklasse fest (DNS funktioniert auch für andere Protokolle als TCP/IP). Für das *Internet* ist dies immer `IN` (INternet).

Typ Definiert den Typ des *Records*. Mögliche Typen sind z.B. `SOA`, `A`, `NS`, `CNAME`, `PTR`, `MX`, `WKS` und `HINFO`.
Beispiel.

```
extor       IN    A  192.168.33.94
```

stellt den Record *für die Domäne* `extor` *dar. Die Gültigkeitsdauer ist nicht angeführt,* `IN` *gibt an, daß es sich um Informationen für das* Internet *handelt,* `A` *definiert die Adresse der gegebenen Domäne (Rechners).*

Daten Die Daten bilden den Informationsinhalt der Initialisierungsdatei. Die Angaben hängen von dem jeweiligen *Typ* ab.

Geht ein *Record* über mehrere Zeilen, so ist die erste Zeile durch eine linke runde Klammer „`(`“ und die letzte durch eine rechte runde Klammer „`)`“ abzuschließen. Kommentare werden wie bei `named.boot` durch einen Strichpunkt begonnen, `*` steht für eine beliebige Anzahl von Zeichen (*Wildcard*).

Beispiel. *Die Zonendatei für die Zone* `firma.de` *könnte folgenden Inhalt haben:*

```
;/etc/namedb/db.firma.de
;Zone firma.de
;Name   ttl  Klasse Typ   Daten
;
@            IN     SOA   extor.firma.de. dnsadm.extor.firma.de. (
                          19970924  ; Version
                          10800     ; Kontaktintervall (3 h) des sekundären Servers
                          1800      ; Neuer Versuch nach einer halben Stunde
                          3600000   ; Gültigkeitsdauer 42 Tage
                          86400     ; Die minimale ttl ist 1 Tag
                          )
             IN     NS    extor.firma.de. ; Autoritative DNS-Server
             IN     NS    ns.extern.de.
;
;Lokale Schleife
;
localhost.   IN     A     127.0.0.1
;
;Mail-Server
;
@            IN     MX    0   extor.firma.de.
             IN     MX    20  mail.extern.de.
;
;Subnetz 2
extor        IN     A     192.168.33.94
             IN     HINFO Pentium/166 Linux
             IN     MX     0  extor.firma.de.
             IN     MX    20  mail.extern.de.
ftp          IN     CNAME extor
mail         IN     CNAME extor
www          IN     CNAME extor
;
intor        IN     A     192.168.33.93
             IN     HINFO Pentium/166 Linux
;
;Subnetz 4
antje        IN     A     192.168.33.129
             IN     HINFO Pentium/166 Linux
```

Der *Record* `SOA` (*Start Of Authority*) beschreibt die Zone, in deren Autorität die nachfolgenden Einträge fallen. Unmittelbar danach folgt der Name des primären DNS-*Servers* dieser Zone. Üblicherweise wird dazu der absolute Name verwendet. Da `@` für die Angabe der impliziten Domäne reserviert ist, wird in der e-mail-Adresse stattdessen ein Punkt verwendet. Die e-mail-Adresse des Verantwortlichen für die oben gegebene Domäne `firma.de` ist folglich `dnsadm@extor.firma.de`. SOA

Hinter der Kontaktadresse sind in runden Klammern fünf Zahlen aufgeführt.

Die erste gibt die Version der Zonendatei an. Bei jeder Änderung der *Records* muß diese Zahl erhöht werden, so daß der sekundäre *Server* erkennen kann, daß sich die Daten geändert haben und er seine aktualisieren muß.

Die zweite Zahl gibt die Dauer der Intervalle an, nach denen der sekundäre *Server* den Inhalt des `SOA`-*Records* kontrolliert. Er vergleicht dabei seine und die Version der Zonendatei des primären *Servers* und aktualisiert gegebenenfalls seine Daten.

Gelingt es dem sekundären *Server* nicht, den primären zu kontaktieren, so wiederholt er seine Versuche in dem Zeitintervall, das durch die dritte Zahl bestimmt wird.

Die vierte Zahl gibt die Anzahl der Sekunden an, nach denen der sekundäre *Server* nach vergeblichen Kontaktversuchen zum primären alle seine Angaben, die er vom primären *Server* über die gegebene Zone besitzt, löscht.

Die letzte Zahl ist die implizite Zeit für den Wert *ttl* für *Records*, die keinen eigenen Wert definieren.

`NS` Der *Record* `NS` (*Name Server*) ist ein Verweis auf den „Nameserver“ der Domäne. Dieser Verweis muß laut RFC 1033 auch in der übergeordneten Domäne enthalten sein. Damit bereits dort eine Überführung der Namen auf IP-Adressen möglich ist, muß in ihr außerdem ein sogenannter *Glue Record* des Typs `A` enthalten sein. Dies ist die Ausnahme von der Regel, daß eine Zonendatei keine *Records* über Zonen enthalten darf, für welche der gegebene DNS-*Server* nicht autoritativ ist.

Beispiel. *In der Zonendatei der Domäne* `de` *muß folglich unter anderem nachfolgender Eintrag stehen:*

```
firma        IN     NS    extor.firma.de.
             IN     NS    ns.extern.de.
extor.firma  IN     A     192.168.33.94
ns.extern    IN     A     172.24.138.1
```

Ohne die beiden letzten Angaben könnte ein Fragesteller zwar den Namen des Servers *für die Domäne* `firma.de` *herausbekommen, könnte ihn aber nicht kontaktieren, da er seine IP-Adresse nicht in Erfahrung bringen würde.*

`MX` Der *Record* `MX` (*Mail eXchanger*) dient zur Weiterleitung der elektronischen Post an einen Mailserver. Die Weiterleitung kann für einen bestimmten Rechner oder eine ganze Domäne erfolgen.

Beispiel. *Eine Nachricht mit der* `To`-*Adresse* ***`benutzer`***`@firma.de` *wird zunächst an den Rechner* `extor.firma.de` *geschickt. Ist dieser nicht zugänglich, so wird sie an den Rechner* `mail.extern.de` *weitergeleitet, wo sie so lange bleibt, bis* `extor.firma.de` *wieder erreicht werden kann. Die Reihenfolge der Weiterleitung wird durch die angegebene Priorität festgelegt. Es wird zunächst versucht, an den Rechner mit der niedrigsten Priorität* `0`*, danach an den Reserverechner* `mail.extern.de` *mit der Priorität* `20`*, weiterzuleiten.*

Ein `MX`-*Record* sollte für alle Rechner, die Post empfangen können, angegeben werden.

Der *Record* `A` (*Address*) ordnet dem Rechnernamen seine IP-Adresse zu. Der dadurch definierte Name wird kanonisch (→ S.25) genannt. `A`

Der *Record* `HINFO` (*Host INFOrmation*) hat rein informellen Charakter. Die beiden folgenden Angaben sind der Typ des Rechners und das verwendete Betriebssystem. Die Konvention für diese Angaben ist in RFC 1340 enthalten. Enthält eine der Angaben Leerzeichen, so muß sie in Anführungszeichen `"..."` gesetzt werden. `HINFO`

Der *Record* `CNAME` (*Canonical NAME*) ordnet dem kanonischen Namen einen Aliasnamen zu. Für den kanonischen Namen muß es einen A-*Record* geben. Aliasnamen dürfen selbst nicht in Zonendateien verwendet werden, sie sind ausschließlich zur externen Verwendung bestimmt. `CNAME`

Aus den im Beispiel 1.4.3 angeführten und bisher noch nicht behandelten Zonendateien verbleiben noch die beiden für die umgekehrten Domänen. Da ihr prinzipieller Aufbau gleich ist, soll hier stellvertretend nur die Datei für die umgekehrte Domäne `33.168.192.in-addr.arpa` erklärt werden.

Beispiel. *Die Zonendatei* `db.192.168.33` *könnte folgenden Inhalt haben:*

```
;/etc/namedb/db.192.168.33
;Domäne 33.168.192.in-addr.arpa
;Name   ttl  Klasse Typ   Daten
;
@            IN     SOA   extor.firma.de. dnsadm.extor.firma.de. (
                          19970924  ; Version
                          10800     ; Kontaktintervall (3 h) des sekundären Servers
                          1800      ; Neuer Versuch nach einer halben Stunde
                          3600000   ; Gültigkeitsdauer 42 Tage
                          86400     ; Die minimale ttl ist 1 Tag
                          )
             IN     NS    extor.firma.de. ; Autoritative DNS-Server
             IN     NS    ns.extern.de.
;
;Subnetz 2
93           IN     PTR   intor.firma.de.
94           IN     PTR   extor.firma.de.
;
;Subnetz 4
129          IN     PTR   antje.firma.de.
158          IN     PTR   intor.firma.de.
```

Der einzige Zweck dieser Datei ist es, den IP-Adressen Namen zuzuordnen. Von allen möglichen *Records* werden daher in der Regel nur `SOA`, `NS` und `PTR` benötigt. `SOA` und `NS` wurden bereits besprochen, es bleibt also nur noch `PTR` übrig.

Der *Record* `PTR` ordnet dem Domänennamen seine IP-Adresse zu. Hinter allen Domänennamen muß dabei ein Punkt folgen, da sonst jedem Namen die implizite Domäne `33.168.192.in-addr.arpa` angehängt würde. Für den Rechner `extor` ergäbe dies z.B. `extor.33.168.192.in-addr.arpa`, was keinen Sinn macht. `PTR`

Wurden in einer der Zonendateien Änderungen vorgenommen, so muß der *Daemon* **named** neu gestartet werden. Je nachdem, wo sich die Datei **named.pid** mit der PID[15] des Prozesses befindet (sie sei z.B. unter **/var/run**), erfolgt der Neustart und damit das erneute Einlesen aller Zonendateien, durch das Kommando

```
extor:~# kill -HUP ‘cat /var/run/named.pid‘
```

Sekundärer Server

In der Initialisierungsdatei **/etc/named.boot** wird für jede Zone, für die der *Server* als sekundär dienen soll, ein Eintrag der Form

`secondary` *Domäne IP-Adresse_des_primären_DNS-Servers Datei*

hinzugefügt.

Beispiel. *Hat die Firma mit dem Betrieb* `betrieb.de` *vereinbart, daß sie für ihn den sekundären* Server*-Dienst sichern wird (der primäre* Server *des Betriebs hat die IP-Adresse* `192.168.42.68`*), so müßte die Datei* **/etc/named.boot** *wie folgt erweitert werden:*

```
directory /etc/namedb
;Typ       Domäne                      Quelle            Reserve
;
primary    firma.de                    db.firma.de
primary    33.168.192.in-addr.arpa     db.192.168.33
secondary  betrieb.de                  192.168.42.68     db.betrieb.de
secondary  42.168.192.in-addr.arpa     192.168.42.68     db.192.168.42
primary    0.0.127.in-addr.arpa        db.127.0.0
cache      .                           db.cache
```

nslookup Zur Kontrolle des DNS-Dienstes dient das Kommando **nslookup**. Es kann zusammen mit dem Domänennamen aufgerufen werden oder ohne. Wird es ohne Argument aufgerufen, so gibt es den DNS-*Server*-Namen aus und fordert den Benutzer mit dem Prompt > zu weiteren Eingaben auf.

Beispiel.

```
extor:~$ nslookup
Default Server:  localhost
Address:  127.0.0.1

> antje
Server:  localhost
Address:  127.0.0.1

Name:    antje.firma.de
Address:  192.168.33.129

>
```

[15] Process Identification Number

Es kann entweder der Name oder die IP-Adresse eingegeben werden. Im letzteren Fall setzt **nslookup** mit Hilfe der Domäne `in-addr.arpa` die Adresse in den Namen automatisch um. Implizit interessiert sich das Programm für die *Records* des Typs `A` und `PTR`. Dies kann durch

`> set type=`*Typ*

geändert werden.

Beispiel. *Angenommen, wir möchten alle Abnehmer der Post für die Domäne* `firma.de` *in Erfahrung bringen.*

```
extor:~$ nslookup
Default Server:  localhost
Address:  127.0.0.1

> set type=MX
> firma.de
Server:  localhost
Address:  127.0.0.1

firma.de   preference = 0, mail exchanger = extor.firma.de
firma.de   preference = 20, mail exchanger = mail.extern.de
firma.de nameserver = ns.firma.de
firma.de nameserver = ns.extern.de
extor.firma.de inet address = 192.168.33.94
mail.extern.de inet address = 172.24.138.3
>
```

Wie man sehen kann, hat der Server *neben den* `MX`-Records *auch die Namen der DNS-*Server *und die IP-Adressen der Mailserver dieser Domäne ausgegeben.*

1.5 IP-Betrieb über serielle Leitungen

Die Protokolle der Familie TCP/IP werden häufig auch auf seriellen Leitungen betrieben. Bei diesen Punkt-zu-Punkt Schaltungen (*Point-To-Point Circuits*) kann es sich um feste Standleitungen oder um Wählleitungen (*Dial-Up Link*) handeln. Der Anschluß der beiden kommunizierenden Rechner erfolgt in der Regel über ihre serielle asynchrone Standardschnittstelle RS-232C und geeignete Modems. Die Modems haben die Aufgabe, die digitalen Signale in analoge und umgekehrt zu wandeln, so daß eine Übertragung über das analoge Telefonnetz möglich ist. Bei digitalen Telefonnetzen, wie z.B. dem ISDN[16], erfolgt der Anschluß über eine ISDN-Karte. Diese ist im Prizip nichts anderes, als ein ISDN-Telefon ohne Hörer mit einer passenden Datenschnittstelle.

Die Schicht der Netzschnittstelle (→ Abb.1.1) wird in der Regel entweder durch das Protokoll SLIP oder PPP realisiert.

[16]Integrated Services Digital Network

1.5.1 SLIP — Serial Line Interface Protocol

Obwohl SLIP kein Standardprotokoll der Familie TCP/IP ist[17], so ist es aufgrund seiner Einfachheit in vielen Kommunikationsprogrammen implementiert und weit verbreitet.

Die grundsätzliche Funktion von SLIP besteht darin, jedem Datagramm ein Byte als Trennzeichen beizugeben und die damit entstehenden Aufgaben zu lösen. SLIP überträgt keine Dienstinformationen, so daß jedem Punkt der Zweipunktverbindung vorher die IP-Adresse seines Gegenübers übermittelt werden muß. Dadurch, daß es nur IP-Protokolle unterstützt, ist es nur beschränkt einsatzfähig.

Zur Initialisierung von SLIP und PPP stehen zwei Programme zur Verfügung — **dip** (*Dialup Interface Protocol*) und **slattach**. Die IP-Adresse des *Clients* kann durch den *Server* statisch oder dynamisch zugeteilt werden.

Statische Adressierung bei fester Verbindung

slattach

Bei fester Verbindung zwischen *Client* und *Server* und statisch zugeteilter Adresse[18] durch den *Server* kann die Initialisierung von SLIP durch **slattach** erfolgen. Dies kann in einer der Intialisierungsdateien oder über die Standardeingabe geschehen.

Beispiel. *Erfolgt der Datenaustausch zwischen* `extor` *und* `verbinde.extern.de` *über eine feste Verbindung, so sind auf* `extor` *folgende Kommandos einzugeben:*

```
extor:~# slattach /dev/modem # Link z.B. nach /dev/ttyS1
extor:~# ifconfig sl0 172.24.233.253 pointopoint 172.24.233.254 netmask 255.255.255.0
extor:~# ifconfig sl0 broadcast 172.24.233.255 mtu 512
extor:~# route add -net 172.24.233.0
extor:~# route add default gw 172.24.233.254
```

Dabei wird auch die IP-Adresse des Gegenübers eingestellt. Mit **mtu 512** *wird die maximale MTU auf 512 Byte eingestellt (→ S.3). Implizit eingestellt ist das Protokoll CSLIP[19]* (Compressed SLIP).

Statische Adressierung bei Wählverbindung

dip

Erfolgt die Anbindung über eine Wählleitung, so muß zusätzlich zu den oben gemachten Angaben das Modem dazu gebracht werden, die richtige Telefonnummer zu wählen, die Verbindung herzustellen und es muß die *Routing*-Tabelle aktualisiert werden. Diese Aufgaben übernimmt **dip**. Das Programm wird durch

dip *skriptname*

gestartet. Die Datei *skriptname* enthält alle notwendigen Anweisungen, um die Verbindung zwischen *Server* und *Client* herzustellen. Diese sind z.B. die Übertragungsgeschwindigkeit, die Initialisierungssequenz für das Modem, die Telefonnummer, der Benutzername, das Passwort und der Start von SLIP.

[17] SLIP wurde bereits 1984 im UNIX der *UC Berkeley*, 4.2 BSD, eingeführt

[18] diese kann zusammen mit dem Namen des *Clients* in **/etc/hosts** eingetragen werden

[19] die „Header“ der IP- und TCP-Felder werden komprimiert, wodurch die Kapazität der seriellen Verbindung erhöht wird

Dynamische Adressierung bei Wählverbindung

Der *Server* teilt in diesem Fall seine IP-Adresse zusammen mit der temporären Adresse des *Clients* gleich bei Aufnahme der Verbindung dem *Client* mit. Die Angaben werden an das Programm **dip** übergeben, wonach es SLIP entsprechend konfiguriert.

SLIP–Server

Die Bedienung der Terminal-Anschlüsse, einschließlich derer, die über ein Modem angeschlossen sind, erfolgt bei UNIX durch **getty**. Nach dem Start von **getty** gibt das Programm eine Aufforderung zum Anmelden (`login:`) aus und startet nach der Eingabe des Benutzernamens eine weitere Prozedur (**login**), die zur Eingabe des Passwortes auffordert.

Die Vorgehensweise ist die selbe, wenn sich ein Benutzer über ein Modem von einem entfernten Rechner aus anmeldet. Er gibt seinen Benutzernamen und sein Passwort an, wonach am *Server* das gewählte Kommunikationsprotokoll gestartet wird.

Bei einem SLIP–*Server* kann als Kommunikationsprotokoll z.B. das Programm **sliplogin** verwendet werden. Es bringt die Terminal-Verbindung in das Regime von SLIP und konfiguriert sie entsprechend der Vorgaben für den sich anmeldenden Benutzer. Diese werden wie folgt getroffen: **sliplogin**

1. Es wird eine Gruppe von Benutzern des Protokolls SLIP eingerichtet, z.B. die Gruppe `slip`.
2. Es werden spezielle Benutzerkonten in der Datei **/etc/passwd** eingerichtet, die als *Login-Shell* das Programm **sliplogin** eingestellt bekommen.
   Beispiel. *Für den Benutzer* `leibner` *kann dieser Eintrag z.B. wie folgt aussehen:*
   `leibner:`*`passwort`*`:111:17:P. Leibner (SLIP):/tmp:/usr/sbin/sliplogin`
3. Des weiteren müssen die Konfigurationsdateien **slip.hosts** und **slip.tty**, sowie die Skriptdateien **slip.login** und **slip.logout** angelegt werden.

In der Datei **slip.hosts** stehen Zeilen der Form

benutzer lokale_IP-Adresse IP-Adresse_Client maske modus zeit

Der erste Eintrag ist der Name des Benutzers, der **sliplogin** gestartet hat. Als nächstes folgt die IP-Adresse des *Servers*, die normalerweise für alle Zweipunktverbindungen die gleiche sein wird[20]. Anstelle von *IP-Adresse_Client* kann die Adresse des *Clients* stehen, die ihm durch den *Server* fest zugeteilt wird, oder das Wort `DYNAMIC`, das eine dynamische Zuteilung der IP-Adresse gestattet. Danach folgt die Subnetzmaske in hexadezimaler Schreibweise. Der Parameter *modus* kann die Werte `normal` für SLIP, `compressed` für CSLIP oder `auto` annehmen. Im letzteren Fall paßt sich die Schnittstelle derjenigen des *Clients* an. Mit dem Parameter *zeit* wird angegeben, wie lange der entfernte *Client* Zeit (in Sekunden) hat, seine Kommunikation zu beginnen, bevor die Verbindung unterbrochen wird (*timeout*).

Beispiel. *Am* Server `verbinde.extern.de` *könnte die Datei* **slip.hosts** *folgende Angaben enthalten:*

[20] ein und dieselbe IP-Adresse kann für alle Zweipunktverbindungen und eine Ethernetschnittstelle verwendet werden

```
leibner  172.24.233.254  172.24.233.253  0xffffff00  auto        60
slip     172.24.233.254  DYNAMIC         0xffffff00  normal      60
cslip    172.24.233.254  DYNAMIC         0xffffff00  compressed  60
```

Für den Benutzer `leibner` *am Rechner* `extor` *wird eine feste Adresse eingestellt, für die anonymen Konten* `slip` *und* `cslip` *erfolgt die Zuteilung der Adresse dynamisch.*

Die Datei **slip.tty** enthält in jeder Zeile ein Wertepaar:

`/dev/tty`*?? IP-Adresse*

Damit wird der seriellen Schnittstelle eine IP-Adresse zugeordnet, die dann dem *Client* mit dem Eintrag `DYNAMIC` in **slip.hosts** nach dem Aufbau der Verbindung zugeteilt wird.

Beispiel.

```
/dev/ttyS0  172.24.233.101
/dev/ttyS1  172.24.233.102
```

Die Dateien **slip.login** und **slip.logout** sind Skriptdateien.

Die erste dient zur Konfiguration der IP-Parameter der neu aktivierten seriellen Schnittstelle. Das Skript **slip.login** wird mit den folgenden Parametern aufgerufen:

`$1`	`$2`	`$3`	`$4`	`$5`	`$6`	`$7`
schnittstelle	*geschwindigkeit*	*PID*	*benutzer*	*lokale_IP-Adresse*	*IP-Adresse_Client*	*maske*

Die *schnittstelle* gibt die symbolische Bezeichnung der Netzschnittstelle wieder, z.B. `sl0`. Die *geschwindigkeit* die Übertragungsgeschwindigkeit und *PID* die PID des Prozesses **sliplogin**. Die folgenden Parameter entsprechen den Angaben aus **slip.hosts**, wobei bei dynamischer Zuteilung der IP-Adresse diese hier verwendet wird.

gated

Werden die SLIP-*Cliets* zu einem Subnetz zusammengefaßt, so muß in den *Routern* dieses Subnetz durch statischen Eintrag oder mit Hilfe des *Daemons* **gated**[21] bekannt gemacht werden. Wird **gated** verwendet, so reicht die Angabe **ifconfig** in **slip.login**, der *Daemon* trägt dann das Subnetz in die *Routing*-Tabelle ein und teilt diesen Weg auch den benachbarten *Routern* mit.

Sind nur wenige *Clients* über Wählleitungen mit dem *Server* verbunden, so kann der *Server* als *Proxy*-ARP (→ S.21) für Rechner, die eine Verbindung aus dem lokalen *Ethernet* zu den SLIP-*Clients* aufbauen wollen, eingerichtet werden.

Beispiel. *Für* `verbinde.extern.de` *könnten die Angaben in der Datei* **slip.login** *wie folgt aussehen:*

```
#!/bin/sh -
/sbin/ifconfig $1 $5 pointopoint $6 netmask $7 mtu 296
/sbin/route add -host $6 dev $1
/sbin/arp -s $6 00:60:8c:df:29:9d pub
exit 0
```

[21] `http://www.gated.merit.edu/`; unterstützt RIP, OSPF, BGP und weitere

Der Aufruf von **slip.logout** erfogt mit den gleichen Parametern wie **slip.login**, es wird lediglich die *PID* weggelassen.

Beispiel. *Die Datei* **slip.logout** *könnte damit folgende Angaben enthalten:*

```
#!/bin/sh -
/sbin/ifconfig $1 down
/sbin/route del $5
/sbin/arp -s $5
exit 0
```

1.5.2 PPP — Point-to-Point Protocol

Gegenüber SLIP hat PPP einige Vorzüge:

- Die Stationen können im Rahmen des Protokolls eine Reihe von Parametern, insbesondere ihre IP-Adressen, vereinbaren. PPP kann in der Schicht der Netzschnittstelle mit Hilfe des Subprotokolls NCP (*Network Control Protocol*) gleichzeitig mehrere Protokolle (z.B. IP und IPX) übertragen, NCP ist daher eigentlich eine Protokollfamilie. Für die Übertragung von IP-Datagrammen und zur Vereinbarung der IP-Adressen wird das Subprotokoll IPCP (*IP Control Protocol*) verwendet. NCP IPCP
- Die gerufene Station (*Server*) kann den Zugang über die Wählleitung wesentlich besser kontrollieren. Dies geschieht mit Hilfe des Subprotokolls LCP (*Link Control Protocol*). Es wird zum Verbindungsauf und -abbau, sowie zur Konfiguration der Verbindungspunkte verwendet. Dazu stehen LCP zwei Methoden zur Verfügung — PAP (*Passwort Authentication Protocol*) und CHAP (*Challenge Handshake Authentication Protocol*). LCP PAP CHAP
- Die Zuverlässigkeit der Übertragung auf einer beliebigen seriellen Verbindung wird durch das Subprotokoll DLLP (*Data Link Layer Protocol*) garantiert. DLLP

Was die Übertragungsgeschwindigkeit betrifft, tritt zwischen den beiden Protokollen kein wesentlicher Unterschied auf.

Eine ähnliche Funktion wie **sliplogin** hat auf der gerufenen Seite **pppd**. Die Beziehungen zwischen rufender und gerufener Seite sind fast symmetrisch, daher ist es oft nicht möglich, die Begriffe *Client* und *Server* zu unterscheiden. Das Programm **pppd** wird auf ähnliche Art und Weise auf beiden Seiten der Verbindung verwendet. Der einzige Unterschied besteht darin, daß auf der rufenden Seite jemand dafür sorgen muß, daß die Telefonnummer gewählt und die Verbindung aufgebaut wird. Diese Aufgabe übernimmt das Programm **chat**. Es verbindet sich mit dem PPP-*Server*, vereinbart mit ihm die IP-Adressen (mit Hilfe von IPCP), konfiguriert die Netzschnittstelle und aktualisiert die *Routing*-Tabelle. **pppd** **chat**

Die Zugangskontrolle erfolgt auf beiden Seiten mittels PAP oder CHAP.

Ähnlich wie bei **sliplogin** kann in der Datei **/etc/passwd** für einen Benutzer als *Login-Shell* **pppd** angegeben werden.

Besteht zwischen den beiden kommunizierenden Rechnern eine feste Verbindung über Telefon oder durch eine direkte Leitung, so entfallen der Verbindungsaufbau und die

Sicherheitskontrollen. Es reicht dann, wenn auf beiden Seiten **pppd** läuft, der z.B. bei der Initialisierung aus einer der Initialisierungsdateien heraus gestartet werden kann.

1.5.3 Automatische Wahl

Normalerweise werden zuerst die Rechner miteinander verbunden (durch Wahl oder fest) und dann erst Daten zwischen ihnen ausgetauscht. Man kann es jedoch auch umgekehrt machen. Die Telefonnummer auf der Gegenseite wird nur dann gewählt, wenn ein Datagramm zu übertragen ist.

diald Diese Aufgabe kann z.B. **diald** übernehmen. Dieser *Daemon* erzeugt eine virtuelle Netzschnittstelle, die sich genauso verhält, wie eine serielle Verbindung mit Hilfe des Protokolls SLIP. Der Systemkern schickt seine Datagramme auf diese Verbindung, die dann von **diald** abgefangen werden. Ist die serielle Verbindung zur Gegenseite nicht aktiv, so versucht **diald** sie zu aktivieren. Ansonsten gibt er die ankommenden Datagramme an die serielle Verbindung weiter, wobei es keinen Unterschied macht, ob diese für SLIP oder PPP konfiguriert ist.

2 TCP/IP-Applikationen

In diesem Kapitel werden einige wichtige Anwendungen der Protokollfamilien TCP/IP, wie die elektronische Post, FTP und das World Wide Web, vorgestellt. Den Beginn macht die sicherlich am weitesten verbreitete Applikation, die elektronische Post.

2.1 Elektronische Post

Die elektronische Post (*e-mail*) ist eine der Grunddienste in Rechnernetzen. Dank *Mail Gateways* funktioniert sie nicht nur im *Internet*, sondern auch zwischen Netzen mit unterschiedlichen Briefformaten und -adressen. Es ist folglich ohne weiteres möglich, einen Brief z.B. aus dem *Internet* in das BITNET, Fidonet, UUCP-Netz oder in Netze, die X.400, DECnet oder andere lokale Postsysteme verwenden, zu schicken. Ein Brief kann gegebenenfalls auch über mehrere solche Netze geleitet werden, ohne daß dabei Informationen verloren gingen.

2.1.1 Formate der Adressen

Jeder elektronische Brief muß mit einer Adresse versehen sein. Soll der Brief auf einen anderen Rechner übertragen werden, so setzt sich die Adresse aus einer Angabe über den Zielrechner und einem lokalen Teil, bei dem es sich in der Regel um den Namen des Benutzers bzw. seines Kontos auf dem Zielrechner handelt, zusammen.

Im *Internet* wird das Format durch RFC 821/822 festgelegt. Es handelt sich dabei um einen Standard, der die Übertragung einer Nachricht in einer 7-Bit ASCII-Kodierung, definiert. Seine Erweiterung auf binäre Dateien und nationale Besonderheiten erfolgte in den RFCs 2045-2049 (MIME). Die allgemeine Form einer *e-mail*-Adresse ist

benutzer@*rechner.domaene*

Für *benutzer* stand in der Regel der Name des Benutzerkontos auf dem Domänenrechner *rechner.domaene*. Heute verwendet man lieber Adressen der Form

Vorname.Nachname@*domaene*

wo statt des Names des Benutzerkontos der Benutzername und statt des Names eines konkreten Rechners nur die Domäne, in dessen Zone der Rechner gehört, angegeben werden. Es ist dabei nicht erforderlich, daß es einen Rechner „*domaene*“ gibt. Der hauptsächliche Vorteil solch einer Adressierung liegt darin, daß beim Wechsel eines Benutzers an einen anderen Rechner innerhalb der gleichen Domäne, keine neue *e-mail*-Adresse vergeben werden muß. Außerdem kann man bei einheitlicher Verwendung dieser Adressierung leichter auch einmal eine *e-mail*-Adresse erraten ohne dazu die interne

Gliederung der Organisation kennen zu müssen und hat zusammen mit der Adresse auch gleich den Namen des Adressaten.

UUCP

Vor der Entstehung des *Internets* wurde zur Übertragung der elektronischen Post UUCP (*Unix-to-Unix Copy*) verwendet. Dort sah die Adressierung wie folgt aus:

rechner1`!`*rechner2*`!`*rechner3*`!`*benutzer*

Das heißt, „schicke den Brief über die Rechner *rechner1* und *rechner2* dem *benutzer* auf dem Rechner *rechner3*“. Wegen des Ausrufezeichens wurde diese Form der Adressen auch *bang-path* genannt. Sie sind relativ. Während des Übergangs über die einzelnen Rechner werden die Adressen des Senders und Empfängers jeweils angepaßt, so daß beim Empfänger die Senderadresse mit den jeweiligen Rechnern, über die der Weg ging, für eine Rückantwort richtig zusammengesetzt vorliegt.

Auch beim absoluten Adressieren im *Internet* ist es möglich vorzugeben, über welche Rechner der Brief geschickt werden soll. Auf diese Art und Weise ist es z.B. möglich, Briefe über ein bestimmtes *Gateway* zu schicken oder vorübergehende Probleme im Netz zu überbrücken. Die allgemeine Form der Adresse ist folgende:

benutzer`%`*zielrechner*`@`*gateway*

Ist das Postprogramm auf dem Senderechner ordentlich konfiguriert, so ist diese Form der Adreßangabe nur selten notwendig. Seinerzeit wurde sie z.B. für Briefe verwendet, die in das englische Netz JANET gingen.

Beispiel. *Solch eine Adresse hatte dann z.B. die folgende Form:*

```
John.Major%downing.gov.uk@cunyvm.cuny.edu
```

Um ein einheitliches und konsistentes Adressierungsschema zu erhalten, wurden sogenannte Pseudodomänen eingeführt. Diese sehen zwar wie normale Domänennamen aus, existieren physikalisch jedoch nicht. Sie dienen lediglich dazu, das Postprogramm bereits auf der Seite des Senders in die Lage zu versetzen, zu erkennen, daß der Brief in ein anderes *e-mail*-System transportiert werden soll und daher die gegebene Adresse zuvor transformiert werden muß. Die wichtigsten Pseudodomänen sind `bitnet` und `uucp`.

Beispiel. *Um einem Benutzer, der im BITNET die Adresse* `KUCKUCK@CSBRMU11` *besitzt, einen Brief zuzustellen, wird folgende Adresse benutzt:*

```
KUCKUCK@CSBRMU11.bitnet
```

Eine richtig konfigurierte Software auf der Seite des Senders sollte folgende Transformation durchführen:

```
KUCKUCK%CSBRMU11.bitnet@earn.cvut.cz
```

Hierbei ist der Rechner `earn.cvut.cz` *das* Mail Gateway*, welches das* Internet *mit dem BITNET verbindet.*

Falls es Probleme mit dem DNS gibt, so kann zur Adressierung des Zielrechners die IP-Adresse auch direkt verwendet werden.

Beispiel. *Wird die IP-Adresse verwendet, so muß sie in eckigen Klammern geschrieben werden, z.B.*

```
Sylvia.Schellberg@[192.168.33.94]
```

2.1.2 Postprogramme

Programme, welche die Zustellung der Post sichern, heißen MTA (*Mail Transfer Agent*), Programme, die zum Lesen und Schreiben der Post dienen, UA (*User Agent*). Der Benutzer hat damit die Freiheit, sich seine UA[1] unabhängig von der verwendeten MTA[2] selbst zu wählen. MTA UA

In Abb.2.1 sind mehrere Möglichkeiten, eine *e-mail* zu empfangen, skizziert.

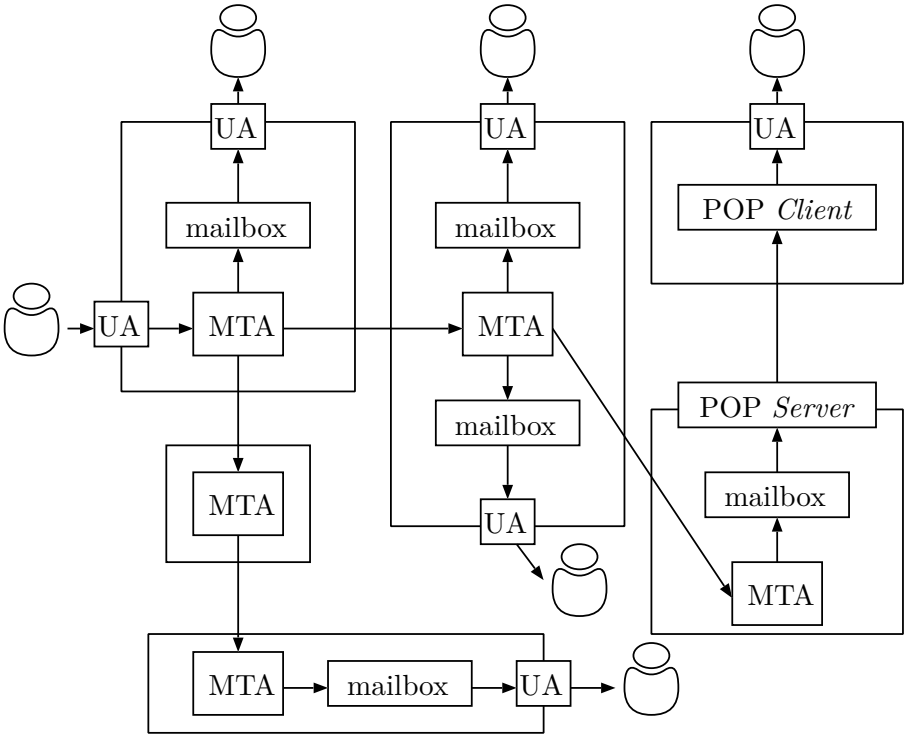


Abb. 2.1: Struktur des *e-mail*-Systems

Die meisten UNIX-Rechner verwenden heute als MTA das Programm **sendmail**. Dieses übernimmt auf der Seite des Absenders die *e-mail* von einer der UAs, stellt eine Verbindung zum **sendmail**-*Daemon* auf der Seite des Empfängers her und übergibt diesem die Post (→ Abb.2.1). Auf der Seite des Empfängers übergibt **sendmail** die Post einem lokalen *Mailer*[3], der sie im Postfach des Empfängers ablegt. Die Postfächer liegen in der Regel unter **/var/spool/mail**. Zur Übergabe der Post zwischen den Rechnern verwendet **sendmail** SMTP (*Simple Mail Transfer Protocol*) [4]. Auch die Kommunikation **sendmail** SMTP

[1] auch MUA (*Mail* UA) genannt, z.B. **pine**, **Zmail**, **elm** oder **mailx**
[2] z.B. **smail3**, **MMDF**, **zmailer**, **PP** oder **sendmail**
[3] z.B. **mail** oder **procmail**
[4] RFC 821, **sendmail** beherrscht auch das ältere UUCP

zwischen UA und **sendmail** erfolgt über SMTP (→ S.51).

Befindet sich das Postfach auf einem anderen Rechner als das Benutzerkonto, so kommt ein weiteres Protokoll ins Spiel (→ Abb.2.1, rechts). Dieses muß dafür sorgen, daß die Post aus dem Postfach auf dem *Mail-Server* abgeholt werden kann. Ein Protokoll, daß
POP diese Funktion erfüllt, ist z.B. POP (*Post Office Protocol*)[5]. Der POP-*Client* holt die Post vom POP-*Server* ab, für abgehende Post wird SMTP verwendet.

SMTP funktioniert nach dem Prinzip *Client/Server*. Der Rechner, der Post empfängt, fungiert als *Server*, Rechner, die ihm Post schicken wollen, stellen mit ihm auf dem
Port 25 TCP-*Port* 25 eine Verbindung als *Client* her.

2.1.3 Form der Briefe

Ein Brief gliedert sich in Umschlag, Briefkopf und -rumpf.

Der Briefkopf[6] setzt sich aus Zeilen zusammen, bei denen links ein Schlüsselwort steht,
Briefkopf das den Inhalt der Zeile bestimmt. Hinter ihm folgt ein Doppelpunkt, ein Leerzeichen und der eigentliche Inhalt der Zeile. Die Reihenfolge der Kopfzeilen ist beliebig. Text in runden Klammern ist ein Kommentar, Text in spitzen enthält Angaben für das Postprogramm (UA).

Beispiel. *Aus der Zeile*

```
Peter Leibner <Peter.Leibner@sta.siemens.de>
```

entnimmt das Postprogramm nur die Angabe, die in spitzen Klammern steht.

Fortsetzungszeilen fangen mit mindestens einem Leer- oder Tabulatorzeichen an, der Kopf ist vom Rumpf durch eine Leerzeile getrennt.

Die wichtigsten Felder des Briefkopfes sind folgende:

`From:` Die *e-mail*-Adresse des Absenders, gegebenenfalls sein bürgerlicher Name.

`Sender:` Es kann sich um die Adresse des Absenders handeln, dann ist sie in der Regel die gleiche wie bei `From` und diese Zeile sollte entfallen, es kann sich aber auch um die Adresse eines Programms handeln. Die Einträge in `From` und `Sender` unterscheiden sich z.B., wenn die Briefe aus einer elektronischen Konferenz stammen. Der Autor (`From`) schickt seinen Beitrag einem Verteilungsprogramm, das ihn dann an die Teilnehmer verschickt. `Sender` sollte in diesem Fall die Adresse des Verwalters dieser Konferenz sein. An die bei `Sender` angegebene Adresse sollten in keinem Fall Antworten gehen, die für die Konferenz bestimmt sind (und umgekehrt).

`Reply-To:` Adresse, an die eine eventuelle Antwort geschickt werden soll. Diese Angabe hat Priorität gegenüber den Angaben in `From` oder `Sender`.

`To:` Die *e-mail*-Adresse des Empfängers, gegebenenfalls auch sein Name.

[5] konkret kann es sich z.B. um den *Daemon* **ipop3d** handeln; ein anderes Protokoll, das etwas umfangreichere Dienste anbietet jedoch weniger verbreitet ist, heißt IMAP (*Internet Message Access Protocol*)

[6] Briefkopf und -rumpf sind in RFC 822 definiert

`Cc:` *Carbon Copy* — Adressen der Empfänger einer Kopie des Briefes.

`BCc:` *Blind Carbon Copy.* Die angeführten Adressaten erhalten eine Kopie des Briefes, diejenigen, die in `To` oder `Cc` aufgeführt sind, erfahren darüber jedoch nichts, da bei ihren Kopien der Eintrag `BCc` fehlt.

`Subject:` Betreff — kurze Angabe über den Inhalt der Nachricht.

`Date:` Wird vom Postprogramm eingetragen. Enthält das Datum und die Zeit (einschließlich Verschiebung zur GMT[7]) des Abschickens der Nachricht.

`Message-ID:` Wird ebenfalls vom Postprogramm hinzugefügt. Sie ist eine eindeutige Identifikationsmöglichkeit für den Brief auf dem Rechner, auf dem er erstellt wurde.

`Received:` Jeder MTA fügt auf dem Weg zwischen Sender und Empfänger (einschließlich) dem Brief Informationen hinzu (→ Tab.2.1).

Bezeichnung	Bedeutung
`by`	Name des Rechners, dessen MTA den Brief angenommen hat
`from`	von wem sie den Brief bekommen hat
`via`	über welches physikalische System der Brief angekommen ist
`with`	mit welchem Protokoll er angekommen ist
`id`	eindeutiger Identifikator des Briefes auf dem Senderechner
`for`	an wen der Brief adressiert war

Tab. 2.1: Elemente der Zeile `Received`

Beispiel.

```
Received: from verbinde.extern.de by extor.firma.de with ESMTP
  id IAA16959 (8.7.5/IDA-1.6 for <Peter.Leibner@firma.de>);
  Wed, 15 Oct 1997 14:52:15 +0200 (MET DST)
```

Zum Zweck der Zustellung durch die MTAs wird der Brief von den UAs der Absender um einen sogenannten Umschlag erweitert. Dieser enthält die Adresse des Absenders sowie eine oder mehrere Empfängeradressen. Der Umschlag ist nicht mit den Einträgen im Briefkopf gleichzusetzen. Die jeweiligen Benutzer erfahren nichts über die dort gemachten Angaben, da sie nach der Zustellung automatisch gelöscht werden. Umschlag

Es folgen einige Gründe, warum es sinnvoll ist, Adressen sowohl im Briefkopf als auch im Umschlag anzubringen.

- Ist der Brief mehreren Empfängern bestimmt, so trennen sich an bestimmter Stelle die Wege der Kopien (→ Abb.2.1). Aus den Umschlägen dieser Kopien werden dabei die Adressen der Briefe entfernt, die einen anderen Weg gingen, im Briefkopf bleiben jedoch alle Adressen (bis auf diejenigen in der `BCc`-Zeile) erhalten.
- Der Benutzer kann mit Hilfe der Datei **.forward** in seinem Heimatverzeichnis vorgeben, wohin der Brief weitergeleitet werden soll. Die Informationen im Briefkopf bleiben dabei unverändert, es wird lediglich die Adresse im Umschlag geändert.

[7] *Greenwich Mean Time*, ist gleich der Angabe UT (*Universal Time*)

- Ist der Brief unzustellbar, so wird der Absender darüber per *e-mail* informiert. Im Kopf steht dann als Absender die Adresse der MTA, an der es zu einem Fehler kam (z.B. `MAILER-DAEMON@`*domaene*). Es kann aber passieren, daß auch die Mitteilung über die Unzustellbarkeit unzustellbar ist. Dann würde eine unendliche Schleife entstehen. Um dies zu verhindern, wird in der Fehlermeldung im Umschlag die Adresse des Absenders weggelassen. Die MTA, die diese Fehlermeldung erhält, unterläßt es dann, eine Fehlermeldung zurückzuschicken.
- Beim Verschicken eines Briefes aus einer *Mailing*-Liste wird die Adresse im Umschlag durch die des Verwalters der Liste ersetzt. Unzustellbare Briefe werden dann dem Verwalter und nicht in die Liste zurückgeschickt. Der Verwalter erhält dadurch die Möglichkeit eine Adresse, die z.B. längere Zeit unzugänglich ist, aus der Liste zu streichen.

Briefrumpf Das ursprüngliche RFC 822 beschränkte das Format des Briefrumpfes auf 7-Bit ASCII-Kodierung und die Anzahl der Zeichen pro Zeile auf 1000. Die ASCII-Kodierung enthält jedoch keine Akzente, geschweigedenn Zeichen z.B. asiatischer Sprachen. Außerdem zeigte es sich, daß die *e-mail* auch zur Übertragung von Binärdateien oder Bildern geeignet ist. Zur Übertragung beliebiger Zeichen bzw. Dateien wurden zunächst Ersatzlösungen gefunden. In UNIX sind es z.B. die Kommandos **uuencode** und **uudecode**. Mit Hilfe von **uuencode** werden binäre Dateien in ASCII-Dateien transformiert, so daß sie per *e-mail* übertragen werden können. Beim Empfänger kann durch **uudecode** wieder die ursprüngliche Form der Datei hergestellt werden.

MIME Eine systematische Lösung bietet MIME (*Multipurpose Internet Mail Extensions*). Da der bestehende Übertragungsmechanismus nicht verändert werden konnte, bestand die Forderung an MIME darin, die ursprüngliche Form des Umschlags und Briefkopfes nicht zu verändern und den Brief so zu kodieren, daß er weiterhin RFC 822 entspricht. Die Aufgabe von MIME besteht folglich darin, den „multimedialen“ (Ton, Bild, Text) Inhalt des Briefes nach RFC 822 und wieder zurück zu transformieren.

Briefkopf Zu diesem Zweck wurden weitere Schlüsselwörter für den Briefkopf eingeführt. Die wichtigsten sind nachfolgend aufgeführt.

`MIME-Version:` Gibt an, daß der Brief entsprechend MIME konstruiert wurde. Derzeitige Version ist `1.1`.

`Content-Type:` Definiert den Inhalt der Nachricht in der Form *typ/subtyp*, wobei *typ* eines der nachfolgenden Schlüsselwörter und *subtyp* eine genauere Spezifikation des *typs* ist.

`text` Einfacher Text. Als *subtyp* kann `plain` oder `rich` vorkommen[8].
Beispiel. *Beim* typ `text` *ist es möglich, die Art der Kodierung anzugeben.*

```
Content-Type: text/plain; charset=us-ascii
```

Die Angabe `charset=us-ascii` *ist der implizite Wert, der immer dann verwendet wird, wenn* `Content-Type` *fehlt. Neben* `us-ascii` *ist noch* `iso-8859-`n *erlaubt, wobei* n *die Ziffer des Kodes ist. Für*

[8] weitere mögliche *subtypen* werden später behandelt

Tschechisch würde die Angabe z.B. `iso-8859-2` *(ISO Latin 2) lauten.*

Bei `rich` handelt es sich um einen Text im Format *richtext.*

`multipart` Dabei handelt es sich um eine Nachricht, die aus mehreren Teilen zusammengesetzt ist. Ein *subtyp* von `multipart` ist z.B. `alternative`, wodurch angegeben wird, daß der Brief denselben Inhalt in unterschiedlichen Formaten (z.B. ASCII-Text und Postscript) enthält.

`application` Binäre Daten, die von Applikationen (z.B. einer Datenbasis oder einem Tabellenkalkulationsprogramm) erzeugt wurden. Als *subtypen* kommen z.B. `octet-stream` oder `postscript` in Frage.

Bei `octet-stream` handelt es sich um „beliebige“ Daten. Damit kann alles mögliche übertragen werden. Die Applikation, die beim Empfänger die Daten verarbeiten soll, wird durch einen Suffix oder einen Begleittext, bestimmt.

Mit `postscript` wird eine Postscript-Datei bezeichnet.

`message` Dient zum Einpacken einer Nachricht. Die *subtypen* sind `rfc822`, `partial` und `external-body`.

Ist der vorliegende Brief nach den Regeln aus RFC 822 zusammengesetzt, so ist der *subtyp* `rfc822`.

Die Angabe `partial` erlaubt die Aufteilung des Briefes in mehrere Teile.
Beispiel.

```
Content-Type: message/partial; number=3; total=7;
 id="<10018.704289854@extor.firma.de>"
```

Mit Hilfe von `id` *wird der Brief eindeutig identifiziert, so daß der Empfänger in der Lage ist, ihn wieder zusammenzusetzen. Der Parameter* `number` *gibt die Nummer des aktuellen Teils,* `total` *die Gesamtzahl der Teile, an.*

Bei `external-body` handelt es sich um einen Verweis. Der Briefrumpf wird durch einen Verweis ersetzt (z.B. auf einen FTP-Server), von dem man die entsprechenden Informationen erhalten kann.
Beispiel. *Ein Verweis auf ein Verzeichnis von Referenzen könnte z.B. wie folgt aussehen:*

```
Content-Type: message/external-body;
 access-type="anon-ftp"; site="ftp.firma.de"
 directory="pub/firma"; name="reference.txt"
```

Durch `access-type` *wird die Art des Zugriffs bestimmt (hier* anonymous ftp*).*

`image` Die Daten repräsentieren ein statisches Bild. Die geläufigsten *subtypen* sind `gif` und `jpeg`.

`audio` Es handelt sich um Tondaten. Der einzige *subtyp* ist `basic`. Dabei handelt es sich um ein Tonsignal, kodiert nach ISDN[9].

`video` Dieser *typ* gibt eine Videosequenz an. Er kann z.B die *subtypen* `mpeg` oder `qtime` haben.

`Content-Transfer-Encoding:` Dieses Schlüsselwort gibt an, mit welchem Algorithmus der Inhalt der Nachricht verarbeitet wurde, um RFC 822 zu erfüllen.

In den RFCs 2045/2046 sind folgende Möglichkeiten aufgeführt:

`7-bit` Dies ist der implizite Wert, der verwendet wird, wenn die Zeile fehlt.

`8-bit` Es handelt sich um kurze Zeilen, in denen Zeichen vorkommen können, die außerhalb des unteren Teils der ASCII-Tabelle (32-126) liegen. Diese Information kann wichtig sein, wenn vor dem Übergang in ein System, das keine 8-Bit Kodierung zuläßt, eine passende Kodierung durchgeführt werden muß. Generell wird nicht empfohlen, diese Kodierung zu verwenden.

`binary` Kodierung für Nachrichten, die in 7-Bit ASCII vorliegen, bei denen jedoch hier und da z.B. ein Bild in 8-Bit Kodierung eingeflochten ist. Sollte ebenfalls nicht verwendet werden.

`quoted-printable` Diese Art der Kodierung wird verwendet, wenn der Großteil der Zeichen aus dem unteren Teil der ASCII-Tabelle stammt, ab und zu jedoch ein Zeichen vorkommt, das speziell behandelt werden muß. Die Kodierung hat den Vorteil, daß auch ohne einer Dekodierung der Text lesbar bleibt.

Die nichterlaubten Bytes werden durch die Zeichen `=`XY ersetzt, wobei XY der hexadezimale Wert des Zeichens in der ASCII-Tabelle ist. Da dadurch das Gleichheitszeichen zu einem Sonderzeichen wird, muß es durch `=3D` ersetzt werden. Zeilen, die im ursprüntlichen Text länger als 76 Zeichen sind, werden auf diese Länge umgebrochen, wobei an das Zeilenende als Zeichen für den Zeilenumbruch (*Soft Line Break*) ein `=` angefügt wird.

`base64` Diese Kodierung ist für allgemeine binäre Daten, die unlesbar sind, geeignet. Es werden dabei jeweils 3 Byte in vier mal sechs Bit aufgeteilt. Jeder dieser vier Gruppen wird dann ein ASCII-Zeichen aus den Mengen `A-Z`, `a-z`, `0-9` und `+` bzw. `/` zugeordnet. Die Größe der Datei wächst damit um ein Drittel. Ist die Größe der Datei nicht ein Vielfaches von 24 Bit, so bleiben am Ende weniger als 24 Bit übrig, also entweder 8 oder 16. Im ersten Fall wird die Anzahl von rechts durch Nullen auf 12 Bit aufgefüllt, die dann mit zwei Zeichen kodiert werden. Hinter diese Zeichen werden noch zwei Gleichheitszeichen angefügt. Im zweiten Fall wird die Anzahl der Bit durch Nullen

[9] *Integrated Services Digital Network*, ein B-Kanal, Abtastfrequenz 8 kHz, mit 8 Bit pro Abtastwert kodiert

auf 18 ergänzt. Hinter die so entstandenen drei Zeichen wird ein Gleichheitszeichen angefügt. Da das Gleichheitszeichen dadurch zu einem Sonderzeichen wird, darf es sonst an keiner anderen Stelle in dieser Kodierung vorkommen.

In RFC 2045/2046 sind noch weitere Kodierungsmöglichkeiten aufgeführt. Sie haben als Schlüsselwort die Bezeichnung `Content-Transfer-Encoding` und müssen mit `x-` (z.B. `x-super-coding`) beginnen.

2.1.4 Routing

Das *Routing* erfolgt im *Internet* anhand der `MX`-*Records* aus den DNS-Zonendateien (→ S.31). Dank dieser Einträge ist es nicht notwendig, daß bei Benutzung von Adressen der Form `MX`-*Records*

Vorname.Nachname`@`*domaene*

ein Rechner namens *domaene* wirklich existiert.

Die ersten beiden `MX`-*Records* auf Seite 31 geben an, daß für die *domaene* `firma.de` (nicht jedoch für ....`firma.de`) die Post an `extor.firma.de` geleitet werden soll (niedrigster Wert der Präferenz). Ist dieser Rechner unzugänglich, so werden sie an `mail.extern.de` geschickt.

Die nächsten `MX`-*Records* geben an, daß Post an die Adresse ...`@extor.firma.de` direkt an den Rechner `extor.firma.de`, bzw., falls dieser unzugänglich ist, an `mail.extern.de` (sekundärer *Mail-Server*), gehen soll. Diese `MX`-*Records* sind (→ S.32) nicht überflüssig. Die meisten MTAs fragen die DNS-*Server* zuerst nach den `MX`-*Records* ab. Die DNS-*Server* fügen dann in der Regel der Antwort die IP-Adressen der Rechner hinzu, zu denen die `MX`-*Records* gehören. Existiert also ein `MX`-Eintrag für den Rechner, so reicht eine einzige Anfrage, falls nicht, so muß die Anfrage nach allen zugänglichen Informationen zu dem gesuchten Namen wiederholt werden.

Eine weitere Möglichkeit, die Post weiterzuleiten, besteht in der Verwendung von Aliasnamen (*Aliases*). Diese Namen werden unter UNIX in einer Datenbasis aufbewahrt. Die Arbeit mit dieser Datenbasis bewerkstelligt **sendmail** mit Hilfe von Funktionen, die in den Standardbibliotheken definiert sind. Es handelt sich dabei um DB bzw. NDBM[10]. Zum Ablegen und Suchen der Daten werden Algorithmen verwendet, die auf sogenannten *Hash Tables* arbeiten. Dank dieser Tatsache ist **sendmail** in der Lage, auch sehr große Datenbasen rasch zu durchsuchen. Aliasnamen

Beispiel. *Die Datei* **/etc/aliases** *könnte z.B. folgenden Inhalt haben:*

```
postmaster: root
players: north, east, south, west, black@chess.org
owner-players: west
```

Erhält **sendmail** *eine Nachricht, in deren Briefkopf als Empfänger ein gültiger Aliasname eingetragen ist, so wird die Nachricht an alle Adressen weitergeleitet, die bei dem Aliasnamen angeführt sind.*

[10] UNIX *Database Management System*, DB (**aliases.db**) ist der Vorgänger von NDBM (**aliases.pa** und **aliases.dir**); Vorsicht, keine der Dateien darf mit **cp** kopiert werden!

Ein Pflichteintrag auf jedem Rechner, der SMTP verwendet, ist der Aliasname **postmaster**.

*Eine andere häufige Verwendung für Aliasnamen sind Gruppenadressen (***players***). Mit jeder solchen Zeile ist der Name eines Besitzers verbunden (***owner-players***). Der Besitzer erhält alle Fehlermeldungen, die im Zusammenhang mit dem Weiterleiten von Nachrichten an die Gruppenmitglieder entstanden sind.*

.forward Die Datei **.forward** stellt eine weitere Möglichkeit dar, die Post weiterzuleiten. Sie wird im Heimatverzeichnis des Benutzers auf dem *Mail-Server* angelegt. In ihr stehen eine oder mehrere Zeilen mit Adressen (*e-mail*-Adressen (lokal, entfernt), Dateien, Programme), an die ankommende *Mails* weitergeleitet werden sollen.

2.1.5 Sendmail

Sendmail entstand zu einer Zeit, als Standards für die elektronische Post gerade erst am entstehen (1981) und als UUCP-Nachrichten oft noch ohne Briefkopf waren. Von Anfang an war es seine Aufgabe, Netze mit unterschiedlichen Post- und Netzprotokollen miteinander zu verbinden. Dies begründet seine weite Verbreitung und seine häufige Verwendung als *Mail-Gateway*.

Sendmail ermöglicht den Empfang und Versand von Post, die lokal, mittels SMTP, UUCP oder anderen Mechanismen, zugestellt werden soll. Die Unterstützung von SMTP ist in **sendmail** integriert, das Programm kann als SMTP-*Client* und -*Server* arbeiten. Zur lokalen Zustellung der Post, für UUCP und andere Mechanismen, ruft **sendmail** externe Programme auf (→ Abb.2.2).

Sendmail ist es gleichgültig, ob mit ihm ein lokaler Benutzer durch sein Postprogramm (z.B. **pine**) oder ein auf einem entfernten Rechner laufender *Sendmail-Daemon* (SMTP) mittels Interprozeßkommunikation in Verbindung tritt. In beiden Fällen erhält es von irgendwo eine Nachricht, die es wieder an ein anderes Programm weitergeben muß. Bei der Weitergabe müssen folgende Leistungen erbracht werden (→ Abb.2.2):

- Verarbeitung der Adressen im Umschlag und Briefkopf. Dazu müssen zuerst alle Adressen der Empfänger gesammelt und eventuelle Aliasnamen oder Adressen aus *Mailing*-Listen expandiert werden.
- Überschreiben der Adressen im Umschlag und Briefkopf, falls der Brief in ein Netz mit einem anderem Adreßformat geschickt werden soll (z.B. UUCP).
- Falls der Brief lokal zugestellt werden soll, dafür sorgen, daß er entweder in das entsprechende Benutzerpostfach oder entsprechend der Angaben in **.forward**, weitergeleitet wird.
- Abschicken des Briefes mit dem ausgewählten Mechanismus (SMTP, UUCP, usw.).
- Falls der Brief unzustellbar ist, den Brief in der Briefschlange ablegen und das Absenden später wiederholen.
- Bei längeren Ausfällen den Brief zusammen mit einer Fehlermeldung an den Absender zurückschicken[11].

[11] Entsteht der Fehler bereits beim Schreiben (UA), so wird der Brief in die Datei **dead.letter** im Heimatverzeichnis des Benutzers abgelegt.

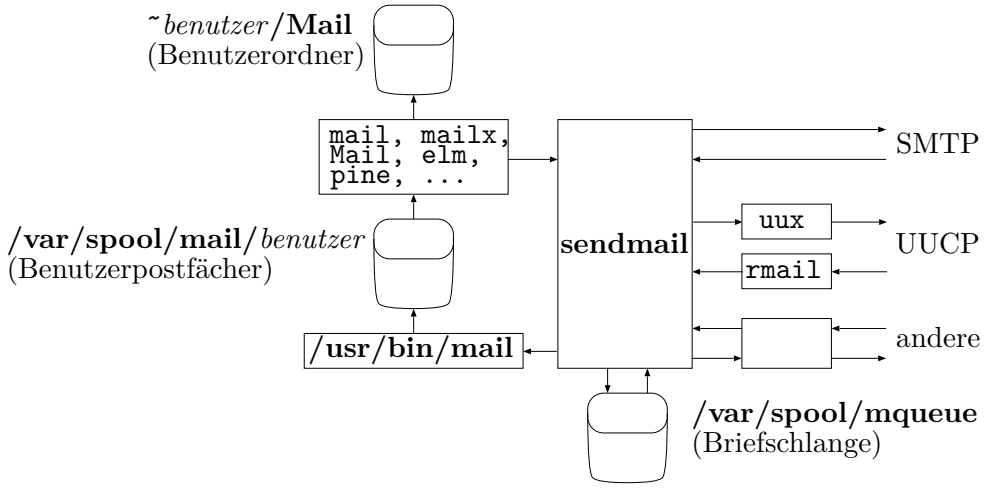


Abb. 2.2: Arbeitsumgebung von **sendmail**

Der Aufbau der Konfigurationsdatei **sendmail.cf** ist so gewählt, daß die oben angegebenen Leistungen erbracht werden können.

2.1.5.1 Konfigurationsdatei sendmail.cf

Unter den angebotenen Netzprogrammen hat **sendmail** eine der kompliziertesten Konfigurationen. Es wird zwar zusammen mit dem Programm eine Beispieldatei mitgeliefert, die dann aber oft durch die Systemadministratoren so stark verändert wird, daß diese selbst nicht mehr verstehen, was sie eigentlich gemacht haben und welche Auswirkungen dies nach sich ziehen wird. Es gibt daher im *Internet* eine ganze Menge schlecht konfigurierter MTAs und viele der Systemadministratoren haben an einen monatelangen Kampf mit **sendmail.cf** die schlimmsten Erinnerungen.

Dazu einige Zitate:

- Falls sie noch nie **sendmail.cf** konfiguriert haben, sind sie kein richtiger Systemadministrator, falls sie es mehr als einmal getan haben, sind sie ein Narr.
- Der Inhalt der Datei **sendmail.cf** gleicht dem Rauschen eines Modems, bzw.
- der Explosion einer Fabrik für Interpunktionszeichen.

Das klassische *Sendmail* benutzt als einzige Konfigurationsdatei **sendmail.cf**. Diese Textdatei kann einige Kilobyte lang sein.

Beispiel. *Zur Abschreckung, ein kleiner Ausschnitt aus* **sendmail.cf***:*

```
#
S13
R$*<@$j>$*        $>3$1$2                 strip local host name
R$*<@$+>$*        $:$>9$1<@$2>$3          canonicalize domain
R$=N              $@$1<@$j>               qualify nonhidden users
#R$-              $:$1??$(@ $1 $: $)      Look up aliases
R$-??$+           $@$1<@$j>               Found it - reformat
R$-??             $:$1                    undo dammage
```

In letzter Zeit wurde die Arbeit des Systemadministrators durch die Aufteilung der Datei in kleinere, leichter überschaubarere, erleichtert. Dies geschah sowohl in der *Sendmail*-Version 8 (und höher) als auch in der IDA-Version von *Sendmail*. In beiden Versionen
m4 werden die Textdateien mit Hilfe des Makroprozessors **m4** in die Datei **sendmail.cf** übersetzt.

Da sich beide Versionen nahezu gleichen, soll stellvertretend für beide kurz die IDA-Version beschrieben werden. Diese Variante ist sehr leicht mit Hilfe von NDBM[12] kon-
dbm figurierbar. Damit wird das *Mail-Routing*, die Art der Überschreibung der Adressen und ähnliches definiert, ohne daß in die Datei **sendmail.cf** eingegriffen werden muß. Bei der Konfiguration durch **m4** wird dann in **sendmail.cf** z.B. eingetragen, welcher Mechanismus (SMTP, UUCP, usw.) und welche Datenbasen verwendet werden sollen. Es entfällt dann die Notwendigkeit, bei einer Änderung der Einträge in der Datenbasis **sendmail** neu starten zu müssen.

Neben der Möglichkeit, Briefe mit Adressen der Form *Vorname.Nachname*`@`... mit Hilfe der Einträge in der Datei **aliases** zu empfangen, bietet IDA-*Sendmail* zusätzlich die Option, die ursprüngliche Form *benutzer*`@`... nach *Vorname.Nachname*`@`... auch bei weggehenden Briefen zu überschreiben. Dieses Überschreiben der Adressen erfolgt entsprechend der Angaben in der Datei **genericfrom**.

Es würde hier zu weit gehen, die Konfiguration von *Sendmail* zu beschreiben. Die Installation eines der beiden Programme sollte jedoch unter Zuhilfenahme der mitgelieferten Dokumentation keine besonderen Schwierigkeiten bereiten `;-)`.

2.1.6 UUCP

In der heutigen Zeit gibt es eigentlich nur zwei Gründe, UUCP als Mechanismus des Postversands und der -zustellung zu verwenden:

1. Es existiert keine Standleitung (diese setzt SMTP[13] voraus) sondern nur eine Wählleitung.
2. Es soll eine MTA (z.B. aus einem entfernten Firmennetz) mit einem *Mail-Server* eines *Providers* über Telefon Post austauschen.

Im Gegensatz zum symmetrischen SMTP, wo jede Seite jederzeit eine Verbindung aufbauen kann, basiert UUCP strikt auf dem Modell *Client/Server*. Es ist immer der *Client*, der eine Übertragung zwischen zwei MTAs, initialisiert.

Die Konfigurationsdateien (→ Tab.2.2) können z.B. unter **/usr/lib/uucp** abgelegt sein. Die Briefschlange könnte z.B. unter **/var/spool/uucp** liegen.

[12] Eintrag in die Datenbasis durch das Kommando **dbm**

[13] eine Standleitung kann jedoch z.B. mit Hilfe von **diald** (→ S.40) vorgetäuscht werden, so daß SMTP verwendet werden könnte — dies wird jedoch wegen des notwendigen häufigen Verbindungsaufbaus (Kosten) nicht empfohlen

Datei	Bedeutung
config	grundlegende Eigenschaften
sys	benachbarte Rechner bzw. Systeme
port	Parameter der Modem-*Ports*
dial	Initialisierungssequenz für die Modems
passwd	Namen und Passwörter

Tab. 2.2: Konfigurationsdateien für UUCP

Um die Funktion von UUCP besser verstehen zu können, soll beispielhaft die Konfiguration von *Client* und *Server* besprochen werden.

2.1.6.1 Client

Ein Rechner `weitweg.firma.de` soll *Mail-Server* für ein abgelegenes Firmennetz sein. Er selbst bezieht seine Post von `extor.firma.de` (Nachbar).

config Der einzige Eintrag in **config** lautet

```
hostname        weitweg
```

Der Name ist für UUCP nicht entscheidend, er könnte genauso gut `nahedran`, lauten.

sys Der einzige Nachbar von `weitweg` ist `extor`. In die Datei **sys** ist daher folgendes einzutragen:

```
system          extor
time            Any
phone           5126388
port            serial1
speed           28800
chat            ogin: weitweg ssword: ujL*p3Qo
alternate
phone           5126391
```

Der Rechner `extor` kann jederzeit angerufen werden (`time Any`), `phone` gibt seine Rufnummer an, `port` bezieht sich auf einen in der Datei **port** definierten Namen und `speed` gibt die Übertragungsgeschwindigkeit an. Falls keine Verbindung mit Hilfe der Standardwerte möglich ist, werden die unter `alternate` (hier eine alternative Rufnummer) angegebenen Werte verwendet.

Bei `chat` (→ S.39) ist der Dialog angeführt, der beim Verbindungsaufbau ablaufen soll. Er lautet: „Wenn du `ogin:` bekommst, schicke `weitweg`, wenn du etwas zurückbekommst, das mit `ssword:` endet, antworte mit `ujL*p3Qo`“

port Die Datei enthält folgende Angaben:

```
port            serial1
type            modem
device          /dev/modem
speed           28800
dialer          us-robotics
```

Der Eintrag `dialer` ist ein Verweis in die Datei **dial**.

dial Diese Datei sieht wie folgt aus:

```
dialer          us-robotics
chat            "" ATZ OK ATDP/T CONNECT
```

`chat` beschreibt den Dialog mit dem Modem. Es soll nicht gewartet werden (`""`), sondern sofort ein ATZ[14] an das Modem erfolgen. Antwortet das Modem mit `OK`, so wird ein ATDP[15] mit der Rufnummer (`/T` wird durch den Wert von `phone` aus der Datei **sys** ersetzt) geschickt. Antwortet daraufhin das Modem durch `CONNECT`, folgt der Verbindungsaufbau entsprechend der Angaben bei `chat` in der Datei **sys**.

Soll der Rechner `weitweg` mit `extor` via UUCP und intern über SMTP kommunizieren, so muß auf ihm *Sendmail* entsprechend konfiguriert werden. Soll intern SMTP verwendet werden, so muß zusätzlich für das isolierte Firmennetz ein DNS-*Server* eingerichtet sein.

Hat z.B. der Postaustausch dreimal täglich zu erfolgen, so sind die jeweiligen Zeiten in die Datei **crontab** des Benutzers **root** einzutragen.

2.1.6.2 Server

Damit der *Server* mit dem *Client* kommunizieren kann, muß auf ihm ebenfalls UUCP laufen. Die Konfigurationsdateien und ihre Bedeutung sind dabei die gleichen wie beim *Client*. Im folgenden sind daher nur die wesentlichen Unterschiede angegeben.

sys Alle Anrufer sollten in der Datei **sys** angegeben werden, jedoch mit dem Unterschied, daß `extor` niemals sie rufen wird. Für `weitweg` würde der Eintrag somit wie folgt aussehen:

```
system          weitweg
time            Never
commands        rmail rnews rsmtp
```

Mit `commands` wird angegeben, welche Kommandos `weitweg` auf dem *Server* ausführen darf.

passwd Gelingt es `weitweg`, eine Modemverbindung herzustellen, so muß sich der entfernte Rechner wie ein gewöhnlicher Benutzer anmelden. Dazu muß für ihn ein entsprechender Benutzer in der Datei **passwd** existieren.

```
weitweg:ujL*p3Qo:333:14:UUCP weitweg:/var/spool/uucp:/usr/sbin/uucico
```

Das Passwort muß demjenigen des Attributs von `chat` in der Datei **sys** auf dem Rechner `weitweg` entsprechen.

Ruft nun `weitweg` an und identifiziert sich mit seinem Namen und Passwort (Dialog entsprechend **chat**), so wird er als Benutzer `weitweg` angemeldet. Für ihn wird dann der Kommandointerpreter **uucico** gestartet. Dieser nimmt mit **uucico** des *Clients* (dieser hat die Verbindung nach seinem Aufruf durch den *Daemon* **cron** initiiert) die Kommunikation auf. Nach erfolgtem Austausch

[14] *ATtention, „Reset“*
[15] *ATtention Dial Pulse*

der Post wird die Kommunikation durch **uucico** wieder beendet und die Verbindung abgebrochen.

2.1.7 Elektronische Konferenzen

Mit Hilfe der *e-mail* ist es möglich, elektronische Konferenzen (*Mailing-Lists*) zu organisieren. Diese Konferenzen haben eine gemeinsame Adresse, über welche Beiträge an alle Teilnehmer verschickt werden können. Die Verwaltung solcher Konferenzen erfolgt ebenfalls mittels *e-mail*. Durch Briefe, die spezielle Kommandos enthalten, können sich die Benutzer z.B. für eine Konferenz anmelden oder wieder aus ihr abmelden bzw. andere Dienste (z.B. eine Teilnehmerliste) anfordern.

Um eine elektronische Konferenz zu installieren, würde es im Grunde genommen genügen, für **sendmail** unter der Adresse der Konferenz eine entsprechende Liste von Aliasnamen einzurichten (→ Bsp.2.1.4). Dadurch würden alle Briefe an diese Adresse an alle Teilnehmer der Konferenz verteilt. Dies würde jedoch die Verwaltung der Liste relativ aufwendig machen und für ihre Abonnenten nicht allzuviel Komfort bieten.

In einer sich dynamisch verändernden Umgebung sollte die Verwaltung der Listen am besten von einem Programm übernommen werden. Solche Programme gibt es unterschiedliche. Als Vorgänger aller gilt LISTSERV, der von IBM für das BITNET entwickelt wurde. Das Programm hat für alle Teilnehmer eine gemeinsame Adresse (häufig `listserv@`*domaene* oder `listproc@`*domaene*) zur Verwaltung der Konferenzen und für jede Konferenz eine eigene Adresse, an welche die Teilnehmer ihre Beiträge schicken können.

Die bekanntesten und am weitesten verbreiteten Programme der Art „LISTSERV“ sind ListProc und Majordomo. Ein weiteres ist TULP, das den Anspruch erhebt, voll kompatibel zum ursprünglichen LISTSERV zu sein.

Um die Arbeitsweise von elektronischen Konferenzen zu veranschaulichen, wird nachfolgend kurz das Programm ListProc, das die meisten Funktionen der erwähnten Programme zusammenfaßt, vorgestellt.

2.1.7.1 ListProc auf der Seite des Servers

ListProc besteht aus einer Reihe von gegenseitig zusammenarbeitenden Programmen. Die Postannahmestelle ist **catmail**. Dieses Programm legt den Brief in das richtige Verzeichnis (z.B. **/net/listproc**) ab.

Die interessanten Unterverzeichnisse von **/net/listproc** sind **requests** (hier werden die Aufträge an die Verwaltung abgelegt) und **lists** (zuständig für die Korrespondenz der laufenden Konferenzen). Die Kontrolle dieser Unterverzeichnisse obliegt dem *Daemon* **serverd**. Sobald dieser eine Anforderung einer Verwaltungstätigkeit (in **requests**) feststellt, startet er das Programm **listproc**, das sich um die Erledigung kümmert. Zur Verteilung der Briefe in die jeweiligen Konferenzen ruft er das Programm **list** auf.

Die Zustellung der Briefe besorgt *Sendmail*. Es muß daher so konfiguriert werden, daß beim Eintreffen eines Briefes in eine Konferenz bzw. für `listproc` oder `listserv` **catmail** aufgerufen wird.

Da **serverd** stets laufen sollte, ist das auf den Start des *Daemons* spezialisierte Programm **start** in eine der Initialisierungsdateien einzutragen.

Die Konfiguration von ListProc erfolgt mit Hilfe von mehreren Dateien. Die wichtigste davon ist **/net/listproc/config**. Sie enthält im ersten Teil die für das gesamte System gemeinsamen Eigenschaften, danach folgen einige Abschnitte mit Eigenschaften der einzelnen Konferenzen. Diese werden durch spezielle Konfigurationsdateien, die sich in den Verzeichnissen der jeweiligen Konferenzen befinden, noch genauer spezifiziert.

Einige der gemeinsamen Eigenschaften sind z.B.

`organization` wonach der Name der Institution folgt,

`manager` mit der Angabe der *e-mail*-Adresse des Verwalters (→ S.46) der *Mailing*-Liste, an den z.B alle Fehlermeldungen (z.B. Unzustellbarkeit) geschickt werden sollen,

`server` mit der offiziellen Adresse für die Administration der Konferenzen (z.B. `listproc@`*domaene*), usw.

Die Angaben zu den Konferenzen gelten jeweils nur für die zu Beginn des Bereichs definierte Konferenz. Die erste Angabe ist folglich immer

`list` wodurch die Konferenz definiert wird. Als erster Parameter folgt der Name der *Mailing*-Liste gefolgt von ihrer *e-mail*-Adresse. Falls die Konferenz moderiert wird, kommt danach die *e-mail*-Adresse des Eigentümers samt Passwort, mit dem er sich zur Administration der Liste anmelden muß. Danach können noch Schalter folgen, die **serverd** beim Aufruf von **list** diesem dann übergibt.

Das Verhalten des Systems kann durch eine Auswahl von Parametern festgelegt werden. Neuen Teilnehmern werden die mit

`default` getroffenen Voreinstellungen zugewiesen.

`default` *konferenz* `{` *parameter*`=`*wert* `}`
Beispiel. *Damit ein Teilnehmer seine* e-mail-*Adresse ändern kann, muß z.B. für die Konferenz namens „tratsch“*

```
default tratsch { address=variable }
```

eingestellt sein.

Eine Zusammenfassung der möglichen Einstellungen enthält Tabelle 2.3.

parameter=wert	Bedeutung
`address=fixed`	der Benutzer kann seine Adresse nicht abändern
`address=variable`	eine Änderung der *e-mail*-Adresse ist auf Wunsch des Benutzers möglich
`mail=ack`	der Benutzer bekommt Kopien seiner Beiträge
`mail=noack`	der Benutzer bekommt keine Kopien seiner Beiträge
`mail=digest`	der Benutzer erhält nur Zusammenfassungen der Beiträge
`mail=postpone`	der Benutzer erhält überhaupt keine Beiträge
`passwort=`*passwort*	Passwort für neue Benutzer; falls kein Wert angegeben wird, so wird ein Passwort automatisch generiert
`conceal=yes`	die Benutzer treten in Statistiken nicht auf
`conceal=no`	die Benutzer treten in Statistiken auf

Tab. 2.3: Parameter und Werte der `default`-Anweisung

ListProc kontrolliert die Köpfe der Briefe. Implizit läßt er nur die Briefe durch, die RFC 822 entsprechen. Sollen auch andere Formate zugelassen werden, so müssen sie durch die Anweisung

`header` hinzugefügt werden.

`header` *konferenz* `{` *briefkopf*`...}`

Daher sollten für Briefe im MIME-Format, die wegen der Beschränkung auf RFC 822 ansonsten ignoriert würden, z.B. folgende Kopfzeilen

`MIME-Version`,
`Content-Type` und
`Content-Transfer-Encoding`,

bei `header` angegeben werden.

Außer den eben erwähnten Anweisungen gibt es noch weitere, wie z.B. `digest`, `ceiling`, `archive` und `disable`, auf die hier jedoch nicht weiter eingegangen werden soll.

Jede Konferenz wird in der Datei **config** definiert. Weitere Informationen zu den Konferenzen liegen verstreut in Dateien, die sich alle unter

/net/listproc/lists/*konferenz*

befinden. Eine Auswahl dieser Dateien ist nachfolgend gegeben.

.subscribers Die Datei enthält Informationen über alle Mitglieder der *Mailing*-Liste. Für jeden Benutzer gibt es eine Zeile der Form

adresse postmodus passwort modus name

Die Zeile beginnt mit der *e-mail*-Adresse des Teilnehmers. Der *postmodus* entspricht einer der Angaben bei `mail` aus Tabelle 2.3. Ist ein *passwort* eingetragen, so wird es bei bestimmten Anforderungen vom Benutzer abgefragt. Für die Angabe *modus* kann `yes` oder `no` angegeben werden. Dadurch wird festgelegt, ob der Benutzer nach außen sichtbar

sein soll oder nicht (→ `conceal` in Tab.2.3). Für *name* ist der Name des Teilnehmers anzugeben.

.news Es besteht die Möglichkeit, die *Mailing*-Liste mit einer *News*-Gruppe zu verbinden. Der Name dieser Gruppe ist dazu in **.news** einzutragen.

.welcome **.welcome** enthält den Text, der allen neuen Teilnehmern einer Konferenz zugestellt wird. Er sollte kurze Informationen über die Absichten der Konferenz sowie die wichtigsten administrativen Angaben (Anmelden, Abmelden) enthalten.

.info Diese Datei enthält eine kurze Einführung in die Thematik der Konferenz. Interessenten erhalten den Inhalt dieser Datei als Antwort auf die Anweisung `information` im Briefrumpf einer *e-mail* an `listproc@`*domaene*.

Ankommende Briefe werden je nach Konferenz entweder in der Datei **mail** oder **moderated** abgelegt. Aus diesen holt sich das Programm **list** die einzelnen Beiträge und verteilt sie nach bestimmten Regeln. Es protokolliert die Ankunft aller Briefe für eine gegebene Konferenz in der Datei **mbox**. Diejenigen Briefe, die es an alle Teilnehmer weitergeleitet hat, kopiert es in die Datei **archive**.

Der bei `list` erwähnte Eigentümer einer Konferenz ist der Hauptbesitzer bzw. Hauptverantwortliche für diese *Mailing*-Liste. Daneben kann es noch mehrere Nebenbesitzer geben. Alle Briefe mit Verwaltungscharakter (Fehlermeldungen) werden ausschließlich an den Haupteigentümer der Liste geschickt. Neben der Angabe des Hauptbesitzers in `list` müssen er und alle Nebenbesitzer zusätzlich in der Datei **/net/listproc/owners** in der Form

e-mail-adresse konferenz

einen Eintrag haben.

ListProc unterstützt zwei unterschiedlich gelenkte Arten von moderierten Konferenzen.

Bei der ersten wird der Brief in die Datei **moderated** im Verzeichnis der Konferenz eingetragen und dem Eigentümer dieser Konferenz eine Nachricht mit der Bitte um Bestätigung geschickt. Dieser kann entweder durch `approve` die Verteilung genehmigen oder durch `discard` den Brief löschen.

Bei der zweiten kann der Haupteigentümer der Konferenz den Brief vor dem Verteilen noch bearbeiten.

Wie bereits erwähnt, besteht auch die Möglichkeit, eine Konferenz mit „Network News“ zu verbinden. Die Verbindung erfolgt durch

`news` *konferenz gruppe e-mail-adresse regime*

Die Übergabe hängt von den Angaben in **config** ab. Steht dort `option post_mail`, so wird die Korrespondenz aus der Konferenz an das Programm **/usr/lib/news/inews** übergeben, das für die Weitergabe in die entsprechende *gruppe* sorgt. In diesem Fall muß der *Server* auch als *News-Server* eingerichtet sein. Bei `gate_mail` erfolgt die Weiterleitung an einen *Gateway*-Rechner mit der gegebenen *e-mail-adresse*.

Mit *regime* wird bestimmt, wie die Übergabe zu erfolgen hat. Bei `receive` ist nur der Empfang von *News*-Artikeln in der Konferenz möglich, bei `send_receive` werden auch Briefe aus der Konferenz in die *News*-Gruppe übernommen.

2.1.7.2 ListProc auf der Seite des Benutzers

Die Kommunikation der Benutzer mit dem System erfolgt durch Briefe an die Adresse `listproc@`*domaene*. Ihre Abarbeitung übernimmt das Programm **listproc**. Jedes Kommando steht in einer eigenen Zeile. Falls **listproc** eine Zeile findet, die mit der Zeichenkette `--` beginnt, nimmt es an, daß danach eine automatische Unterschrift folgt und ignoriert den Rest.

Kurzinformationen zu den einzelnen Kommandos erhält man durch die Angabe `help` (`help` *kommando* liefert Informationen zu einem speziellen Kommando) im Briefrumpf einer *e-mail* an `listproc@`*domaene*.

Zum An- und Abmelden dienen die Kommandos

`subscribe` *konferenz teilnehmer*
`unsubscribe` *konferenz*
`signoff` *konferenz*

Das erste dient zum Anmelden des Benutzers *teilnehmer* in die Liste *konferenz*. Für *teilnehmer* ist der bürgerliche Name anzugeben. Über die Mitgliedschaft entscheidet die *e-mail*-Adresse, die das Programm automatisch aus dem Briefkopf feststellt. Des weiteren legt das Programm ein Passwort fest, das zum interaktiven Zutritt (von dem hier nicht gesprochen wird) und zur Abänderung der eigenen *e-mail*-Adresse benötigt wird.

Die Kommados `unsubscribe` und `signoff` sind Synonyme. Beide führen zum Erlöschen der Mitgliedschaft in *konferenz*. Mit `which` kann man in Erfahrung bringen, in welchen Konferenzen man angemeldet ist. Ein Verzeichnis aller Konferenzen erhält man mit `lists`.

Informationen zu den Konferenzen liefern mehrere Kommandos. Als Parameter haben sie jeweils den Namen der Konferenz (z.B. `tratsch`). Mit `recipients` oder `review` erhält man ein Verzeichnis aller Mitglieder (außer von denen, die `conceil=yes` eingestellt haben) einer Konferenz. Allgemeine Informationen zur Konferenz liefert `information`, `statistics` schickt Angaben zur Anzahl der Beiträge einzelner Konferenzteilnehmer.

Mit `set` können die Parameter verändert werden, die das Verhalten des *Servers* dem Teilnehmer gegenüber bestimmen.

`set` *konferenz parameter wert*

Die einzelnen Werte entsprechen den Angaben aus Tabelle 2.3. Das Kommando `set` unterscheidet sich gegenüber dem `set` in **config** auf zweierlei Art und Weise — statt dem Gleichheitszeichen wird hier ein Leerzeichen verwendet und bei `address` kann die *e-mail*-Adresse angegeben werden, die für den Teilnehmer neu einzustellen ist (→ Bsp.2.1.7.1).

Mit `index` kann man ein Verzeichnis von Dateien aus dem Archiv einer Konferenz erstellen. Mit `get` kann ein ausgewählter Beitrag aus dem Archiv geholt und mit `search` kann ein Archiv nach bestimmten Dateien durchsucht werden. Dabei sind für die Dateinamen reguläre Ausdrücke, so wie sie für **egrep** gelten, erlaubt. Genauere Angaben zu diesen Kommandos können mit `help` *kommando* direkt beim *Server* abgeholt werden.

Die Version von ListProc erhält man mit `release`.

2.2 FTP — File Transfer Protocol

FTP ist das Standardprotokoll des *Internets* zur Übertragung von Dateien. Der Benutzer stellt dazu zuerst eine Verbindung zwischen zwei Rechnern her und überträgt dann mit Hilfe von FTP-Kommandos die Dateien von dem einen Rechner auf den anderen. Im Gegensatz dazu erfolgt die Übertragung von Dateien bei NFS ohne das der Benutzer vorher eine Verbindung zwischen den beiden Rechnern herstellen muß[16].

Port 21 Der *Client* stellt zu Beginn der Sitzung eine Verbindung mit dem *Server* auf *Port* 21 her. Diese Verbindung dient ausschließlich zur Steuerung der Übertragung und bleibt während der gesamten Sitzungsdauer bestehen.

Port 20 Zur Datenübertragung wird eine weitere Verbindung auf *Port* 20 hergestellt. Diese besteht nur solange, bis die Datenübertragung (z.B. Übertragung einer Datei oder Ausgabe des Verzeichnisinhalts) beendet ist.

Abb. 2.3: Übertragungskanäle bei FTP

Das Ende der Sitzung wird meistens durch den *Client* mit den Kommandos **bye** oder **quit** eingeleitet. Diese beenden die Steuerverbindung, wonach alle am *Server* gespeicherten Zustandsinformationen gelöscht werden, so daß bei einem erneuten Anmelden des *Clients* keine Informationen mehr über eine frühere Verbindung existieren (→ Abb.2.3).

Es ist zu beachten, daß Steuerbefehle nicht gleich *Client*-Befehle sind. So dient z.B. zur Ausgabe des Verzeichnisinhalts der Steuerbefehl **LIST**, am *Client* wird dazu der Benutzer jedoch in der Regel **dir** eingeben. Die richtige Umsetzung des Kommandos in einen Steuerbefehl besorgt im *Client*-Programm ein Kommandointerpreter. Dank dieser Eigenschaft können für den *Client* eigene Befehle zur Interaktion mit dem Benutzer definiert werden.

2.2.1 FTP — Server

FTP ist in den meisten Versionen von UNIX bereits vorinstalliert. Der bekannteste FTP-*Server* dürfte wohl derjenige der Washington University in Saint Louis (**ftpd**)

wu-ftpd sein. Er ist frei verfügbar und im *Internet* unter dem Namen **wu-ftpd**[17] zu finden. Der Start des FTP-*Servers* erfolgt durch den *Daemon* **inetd** (*internet daemon*).

> Auf TCP/IP-Stationen werden in der Regel mehrere Netzdienste (z.B. Telnet, FTP, SMTP, Talk, usw.) angeboten. Die Stationen treten für manche gleichzeitig als *Client* und *Server* auf. Die Funktion des *Servers* eines jeden

inetd solchen Dienstes kann durch den „Superdaemon“ **inetd** realisiert werden. Er

[16] ein Teil des Dateisystems eines anderen Rechners wird dazu in das Dateisystem des eigenen integriert (*mounted*), wodurch der Benutzer den Eindruck gewinnt, er greife auf lokale Daten zurück, obwohl diese zu ihm über das Netz (in der Regel mittels UDP) übertragen werden

[17] eine Alternative für Linux ist **troll-ftpd**, der zwar nur einen Teil von RFC 959 abdeckt, aber dafür sehr einfach und sicher ist

hat die Funktion eines Vermittlers, der an mehreren Standardports lauscht und beim Eintreffen einer Anforderung den zuständigen *Server*-Dienst startet.

Es liegt im Ermessen des Systemverwalters, welche Dienste dauerhaft und welche nur nach Aufforderung gestartet werden. So werden z.B. die *Daemonen* **named** und **sendmail** (und natürlich **inetd** selbst) oft permanent (in einer der Initialisierungsdateien, z.B. **rc.config**) gestartet, wogegen FTP z.B. zu den Diensten gehören kann, die nur nach Bedarf „aufgeweckt“ werden sollen.

Die Konfiguration von **inetd** erfolgt mit Hilfe der Datei **/etc/inetd.conf**.
Beispiel. *Nachfolgend ist ein kleiner Ausschnitt aus der Datei* **inetd.conf** *gegeben. Dabei können die angegebenen Einträge z.B. so,*

```
ftp    stream tcp nowait root   /usr/sbin/ftpd ftpd
telnet stream tcp nowait root   /usr/sbin/telnetd telnetd
tftp   dgram  udp wait   nobody /usr/sbin/tftpd tftpd
```

oder so

```
ftp    stream tcp nowait root   /usr/sbin/tcpd ftpd
telnet stream tcp nowait root   /usr/sbin/tcpd telnetd
tftp   dgram  udp wait   nobody /usr/sbin/tcpd tftpd
```

aussehen. Die ersten drei Zeilen zeigen einen normalen Ausschnitt aus der Datei **inetd.conf**, *in den letzten drei Zeilen wird bei allen Applikationen der TCP-*Daemon (TCP Wrapper) **tcpd** *verwendet, der es ermöglicht, den Zugang zu den einzelnen Diensten in Abhängigkeit von der Domäne, von der aus sich der Benutzer anmeldet, zu kontrollieren (zu erlauben oder zu verweigern).* **tcpd**

Von links nach rechts betrachtet wird in den Zeilen zuerst der Dienst (gemäß dem Eintrag in **/etc/services***), dann der Typ (*`stream` *für verbundene Dienste, die TCP verwenden,* `dgram` *für nichtverbundene, die UDP benutzen und* `raw` *für IP-Datagramme) gefolgt vom Protokoll (aus* **/etc/protocols**, *meistens* `tcp` *oder* `udp`*), dem Schlüsselwort* `wait` *oder* `nowait` *(bei* `wait` *wartet* **inetd**, *bis der* Server *die* Client*-Anforderung am zugehörigen* Port *erledigt hat, bei* `nowait` *wird der* Client *auf einen dynamisch zugeteilten* Port *umgelegt und* **inetd** *„lauscht“ sofort wieder an dem frei gewordenen* Port*), dem Namen des Benutzers, mit dessen Zugriffsrechten das* Server*-Programm gestartet wird, dem vollständigen Weg zum* Server*-Programm und die Argumente, die dem* Server *übergeben werden sollen (das erste ist immer* `argv[0]`, *also der Programmname bzw.* `internal`, *für interne Dienste), angeführt.*

Konfiguriert wird **ftpd** durch mehrere Konfigurationsdateien (→ Tab.2.4) und durch die Angabe von Schaltern bei seinem Aufruf.

Datei	Bedeutung
ftpaccess	legt die Benutzerkategorie, ihre Möglichkeiten und weitere Parameter fest
ftpconversions	automatische Formatwandlung, die der *Server* durchführen kann
ftpusers	Verzeichnis von (nichtanonymen) Benutzern, die FTP *nicht* benutzen dürfen (da über das Netz Passwörter unverschlüsselt geschickt werden, wird z.B. der Zugang für **root** meistens gesperrt)
ftpgroups	Benutzergruppen für FTP
ftphosts	von welchen Rechnern sich welche Benutzer anmelden dürfen

Tab. 2.4: Konfigurationsdateien für FTP

2.2.1.1 Anonyme FTP-Server

Benutzer, die auf einem bestimmten Rechner ein Benutzerkonto haben, können FTP zur Übertragung von Dateien in beide Richtungen nutzen. Nach der Eingabe von **ftp** geben sie ihren Benutzernamen und ihr Passwort an, so als ob sie sich direkt am Terminal anmelden würden.

Neben dieser Möglichkeit bieten viele Rechner auch anonymen Benutzern Zugang. Diese Benutzer melden sich entweder unter **ftp** oder **anonymous** am System an.

Da der anonyme Zugang eine Gefährdung des Systems bedeutet, müssen die Zugriffsrechte sorgfältig vergeben werden. Für den anonymen Benutzer sollte eine spezielle Gruppe, z.B. `anonftp`, vergeben werden, deren einziges Mitglied `ftp` sein wird. Der Eintrag in **/etc/group** könnte dann

```
anonftp::19:
```

lauten (die GID[18] sei 19). Weiter ist es ratsam, den anonymen FTP-*Server* in ein eigenes Verzeichnis, am besten auf einer getrennten Partition bzw. einer eigenen Festplatte, einzurichten[19]. Dieses könnte z.B. **/net/ftp** sein. Für den Benutzer lautet der Eintrag in **/etc/passwd** dann z.B. wie folgt:

```
ftp:*:404:19:Anonymous FTP user:/net/ftp:/bin/false
```

Das Passwort des Benutzers `ftp` ist blockiert (kein Passwort entspricht `*`) und als *Login-Shell* steht `/bin/false`, was zu einem sofortigen Abbruch führt, wenn der Benutzer versucht, sich vom Terminal oder über Telnet anzumelden. Die anonymen Benutzer haben nur Zugang zu den Dateien im Heimatverzeichnis[20] des Benutzers `ftp`[21]. Das Verzeichnis **/net/ftp** sowie alle darunter liegenden Verzeichnisse sollten dem Benutzer `root` und der Gruppe `ftp` gehören.

[18] Group Identification Number

[19] diese muß in **/etc/fstab** durch `mount...` in das Dateisystem integriert werden

[20] der *Daemon* **ftpd** ruft nach dem Anmelden des anonymen Benutzers **chroot** auf, wodurch `/net/ftp` zum Wurzelverzeichnis für den anonymen Benutzer gemacht und ihm der Zugriff auf andere Verzeichnisse außerhalb von `/net/ftp` verwehrt wird

[21] das Heimatverzeichnis soll hier durch **~ftp** gekennzeichnet werden

Die unter **/net/ftp** liegenden Verzeichnisse sind in der Regel folgende:

bin Nachdem das Wurzelverzeichnis mit **chroot** zu **/net/ftp** gemacht wurde, sind dem anonymen Benutzer alle außerhalb dieses Verzeichnisses liegenden Kommandos nicht mehr zugänglich. Es wird daher ein Verzeichnis **/net/ftp/bin**[22] angelegt und in dieses die Kommandos **ls**, **compress**, **uncompress**, **gzip**, **gunzip** und **tar** kopiert. Dem Verzeichnis und den darin enthaltenen Dateien ist als Eigentümer `root.ftp` einzustellen und die Zugriffsrechte sind auf `---x--x--x` bzw. `d--x--x--x` (111) zu setzen.

etc Damit **ls** die Benutzer- und Gruppennamen bei der Ausgabe (**ls -l**) findet, müssen **/etc/passwd** und **/etc/group** nach **/net/ftp/etc** kopiert werden. In den Kopien sollten nur die Zeilen der Eigentümer von Dateien unter **~ftp** belassen werden. Des weiteren sollte bei ihnen das Passwort durch einen `*` blockiert werden.

In **/net/ftp/group** steht als einzige Gruppe `ftp`.

Der Eigentümer von **etc** und den beiden Dateien ist wieder `root.ftp`, die Zugriffsrechte für die Dateien sind `-r--r--r--`, für **etc** `d--x--x--x`.

pub Für den anonymen Benutzer ist das Verzeichnis **/net/ftp/pub** sicherlich das interessanteste. Auch hier ist der Besitzer `root.ftp`. Die Zugriffsrechte sollten für **pub** auf `dr-xrwsr-x` [23] (`-rw-rw-r--`, `drwxrwxr-x`) gesetzt werden.

Falls dem anonymen Benutzer überhaupt ein Schreibrecht eingeräumt werden soll, so sollte dafür ein eigenes Verzeichnis reserviert werden, z.B. **/net/ftp/incoming**. Der Eigentümer ist `root.anonftp`, die Zugriffsrechte `drwx-wx---`. Der anonyme Benutzer hat in diesem Verzeichnis zwar das Schreibrecht, er kann den Inhalt des Verzeichnisses jedoch nicht ausgeben (das Leserecht fehlt).

2.2.1.2 Konfigurationsdateien ftpaccess und ftpconversions

Von den in der Tabelle 2.4 angegebenen Konfigurationsdateien seien hier nur die beiden wichtigsten erwähnt.

In **/etc/ftpaccess** werden Benutzer- und Rechnerklassen sowie deren Zugriffsrechte definiert und Aspekte der Sicherheit festgelegt.

Beispiel. *Zur Veranschaulichung soll der Inhalt einer konkreten Datei* **ftpaccess** *durchgenommen werden.*

```
1  loginfails 2
2  class   local   real,guest,anonymous *.firma.de
3  class   remote  real,guest,anonymous *
4  limit   local   20  Any                /etc/msgs/msg.toomany
5  limit   remote  100 SaSu|Any1800-0600  /etc/msgs/msg.toomany
6  limit   remote  60  Any                /etc/msgs/msg.toomany
7  readme  README*     login
```

[22] da **/net/ftp** das Wurzelverzeichnis ist, liegt **bin** aus Sicht des anonymen Benutzers unter **/bin**

[23] **chmod 2575 pub** setzt für die Gruppe das Set-GID-Bit, was zur Folge hat, daß alle unter **pub** erzeugten Unterverzeichnisse und Dateien automatisch der Gruppe `ftp` angehören werden

```
 8  readme  README*     cwd=*
 9  message /welcome.msg            login
10  message .message                cwd=*
11  compress        yes             local remote
12  tar             yes             local remote
13  private         yes
14  passwd-check    rfc822  warn
15  log commands    real
16  log transfers   anonymous,real inbound,outbound
17  shutdown /etc/shutmsg
18  delete          no              guest,anonymous
19  overwrite       no              guest,anonymous
20  rename          no              guest,anonymous
21  chmod           no              anonymous
22  umask           no              anonymous
23  upload  /net/ftp    *           no
24  upload  /net/ftp    /pub        no
25  upload  /net/ftp    /bin        no
26  upload  /net/ftp    /etc        no
27  upload  /net/ftp    /incoming   yes root daemon 0600 dirs
28  alias   inc:        /incoming
29  cdpath  /incoming
30  cdpath  /pub
31  cdpath  /
32  path-filter anonymous /etc/pathmsg ^[-A-Za-z0-9_\.]*$ ^\. ^-
33  path-filter guest     /etc/pathmsg ^[-A-Za-z0-9_\.]*$ ^\. ^-
34  guestgroup ftponly
35  email Peter.Leibner@firma.de
```

In der Zeile 1 *werden jedem* Client *zwei Versuche erlaubt, sich richtig anzumelden. Falls er sie verbraucht, so wird die Verbindung abgebrochen.*

In der Zeile 2 *wird die Klasse lokaler Benutzer definiert (*`local`*). In sie gehören alle drei möglichen Kategorien von Benutzern —* `real`*,* `anonymous` *und* `guest`*. Der ersten gehören alle Benutzer an, die auf dem Rechner ein eigenes Konto haben, die Bedeutung der zweiten ist klar und* `guest` *ist ein Zwischending zwischen den beiden anderen. In der Datei wird sie durch* `guestgroup` *(Zeile* 34*) definiert. Der letzte Eintrag ist* `*.firma.de`*. Damit werden die Rechner der Zone* `firma.de` *als* `local` *definiert. Das bedeutet unter anderem, daß ein Rechner, von dem aus sich der* Client *anmeldet, einen Eintrag in der Zonendatei der umgekehrten Domäne haben muß. An dieser Stelle könnte aber auch* `192.168.33.*` *stehen, wodurch die lokalen* Client*-Rechner auf das angebene IP-Netz eingeschänkt würden.*

In Zeile 3 *wird analog die Klasse* `remote` *definiert, in die alle anderen gehören, woher sie sich auch immer anmelden mögen.*

In Zeile 4 *wird die Anzahl der* Clients *auf 20 beschränkt.* `Any` *besagt, daß diese Einschränkung alle Tage zu jeder Zeit gilt. Der letzte Eintrag gibt die Datei an, deren Inhalt einem überzähligen* Client *geschickt wird. Den* Clients*, die in die Klasse* `remote`

gehören, sind für Samstage und Sonntage oder nach Feierabend (18-6h) bis zu 100 Anmeldungen gestattet (Zeile 5*), außerhalb dieser Zeit sind es 60 (Zeile* 6*).*

Die nächsten beiden Zeilen, 7 *und* 8*, definieren sogenannte „magische" Dateien. Dem* Client *wird nach dem Anmelden (Zeile* 7*) bzw. nach einem Verzeichniswechsel (Zeile* 8*) mitgeteilt, wann die angegebenen Dateien (im Beispiel Dateien, die mit* `README` *beginnen) zuletzt geändert wurden.*

In den Zeilen 9 *und* 10 *wird eine weitere Gruppe „magischer" Dateien definiert. In diesem Fall wird nach dem Anmelden (Zeile* 9*) bzw. nach einem Verzeichniswechsel (Zeile* 10*) der Inhalt der angegebenen Dateien dargestellt. Der Inhalt von* `.message` *kann nur dann ausgegeben werden, wenn solch eine Datei in dem Verzeichnis, in das gewechselt wurde, auch existiert.*

Die Zeilen 11 *und* 12 *gestatten den angegebenen* Clients *die Programme zur Dateikompression und Archivierung zu nutzen. Konkrete Instruktionen für solches implizites Bearbeiten zu übertragender Dateien sind in der Datei* **ftpconversions** *enthalten.*

Durch die Angabe in Zeile 13 *wird dem* Client *erlaubt,* **SITE GROUP** *und* **SITE GPASS** *des FTP-Protokolls zur Erweiterung seiner Zugriffsrechte anzuwenden.*

Die Anweisung `passwd-check` *in Zeile* 14 *bestimmt, wie genau das Passwort des anonymen Benutzers zu sein hat. Im Beispiel wird der Benutzer bei der Eingabe eines falschen Passwortes gewarnt und darauf hingewiesen, als Passwort seine* e-mail-*Adresse nach RFC 822 zu verwenden, danach wird er aber vom* Server *akzeptiert. Wird anstelle von* `warn enforce` *verwendet, besteht der* Server *auf der Eingabe des richtigen Passwortes, sonst bricht er die Verbindung ab. Anstelle von* `rfc822` *kann auch* `none` *(beliebiges Passwort) oder* `trivial` *(es wird nur kontrolliert, ob das Passwort ein* `@` *enthält), stehen.*

Die nächsten beiden Zeilen bestimmten den Umfang der Informationen, die mittels des Daemons **syslogd** *von dem* Client *gespeichert werden. Zeile* 15 *gibt an, daß die Kommandos ordentlicher Benutzer (*`real`*) komplett gespeichert werden, Zeile* 16*, daß die Übertragung von Dateien für* `real` *und* `anonymous` *in beide Richtungen (*`inbound` *und* `outbound`*) registriert wird.*

Die Anweisung `shutdown` *in Zeile* 17 *ermöglicht es, mit Hilfe von Einträgen in der speziellen Datei* **/etc/shutmsg** *bereits angemeldete Benutzer auf ein bevorstehendes Herunterfahren des* Servers *aufmerksam zu machen bzw. neu sich anmeldende Benutzer ab einem bestimmten Zeitpunkt vor dem geplanten „Shutdown" nicht mehr zuzulassen.*

Die folgenden fünf Zeilen (18-22) *bestimmen, welche Operationen die angegebenen* Clients *mit den Dateien auf dem* Server *durchführen dürfen.*

Die Anweisungen `upload` *in den Zeilen* 23-27 *beschränken für die anonymen Benutzer die Möglichkeiten, Dateien auf dem* Server *abzulegen. Sie betreffen diejenigen Benutzer (*`ftp`*), deren Wurzelverzeichnis* `/net/ftp` *ist. Diese dürfen weder in* `~ftp`*,* `~ftp/pub`*,* `~ftp/bin` *noch in* `~ftp/etc` *Dateien ablegen, in* `~ftp/incoming` *ist es ihnen dagegen erlaubt. Dort abgelegte Dateien erhalten den Benutzer* `root` *und die Gruppe* `daemon` *zugeordnet. Ihre Zugriffsrechte sind* `-rw-------` *(0600). Des weiteren ist es ihnen gestattet, in* `~ftp/incoming` *Unterverzeichnisse anzulegen (*`dirs`*).*

In Zeile 28 *wird der Aliasname* `inc:` *für* `/incoming` *definiert. Diese Abkürzung kann dann im Kommando* **cd** *(z.B.* **cd inc:***) verwendet werden.*

In den Zeilen 29-31 *stehen diejenigen Verzeichnisse, in denen* **cd** *nach Unterverzeichnissen sucht. Bei der Angabe von z.B.* **cd linux** *wird* **wu-ftpd** *nacheinander versuchen, das Arbeitsverzeichnis nach* **./linux**, **/incoming/linux**, **/pub/linux** *oder* **/linux** *zu verlegen.*

Mit `path-filter` *wird für die gegebenen Benutzer die Wahl der Dateinamen eingeschränkt. Die angegebenen regulären Ausdrücke schränken die Namen auf Groß- und Kleinbuchstaben sowie Ziffern und die Zeichen* `_`, `-` *und* `.` *ein und sie dürfen nicht mit einem* `-` *oder* `.` *beginnen. Auf fehlerhafte Eingaben wird mit einer Meldung aus* **/etc/pathmsg** *reagiert.*

In der Zeile 34 *wird die Kategorie* `guest` *definiert, der alle Benutzer der Gruppe* `ftponly` *angehören werden.*

Die letzte Zeile gibt die e-mail-*Adresse des Administrators des FTP-*Servers *an.*

Sehr nützliche Dienste kann die automatische Konversion von Dateien leisten. Dazu benötigt der *Server* die Angaben in den Zeilen 11 und 12 der Datei **ftpaccess** und die Datei **ftpconversions**. Die Syntax dieser Datei besteht aus acht durch Doppelpunkt getrennten Einträgen pro Zeile:

1. Zu entfernender Präfix – bisher nicht genutzt.
2. Zu entfernender Suffix. Dieser Anhang soll vom Namen entfernt werden.
3. Hinzuzufügender Präfix – bisher nicht genutzt.
4. Anzuhängender Suffix. Durch die Angabe des zu entfernenden Anhangs und des neue hinzuzufügenden kann die Datei von einem Format in ein anderes überführt werden.
5. Das Kommando, das zum Entfernen oder Anhängen ausgeführt werden soll. Spezielle Bedeutung kommt dabei der Zeichenkette `%s` zu, die vor jeder Ausführung des Kommandos durch den Dateinamen ersetzt wird.
6. Typen.

7./8. Auswahl.

Die letzten drei Einträge sind in der Dokumentation leider nicht beschrieben, die obigen Angaben sind daher nur geraten.

Beispiel. *Als Beispiel sei eine konkrete Datei*[24] **ftpconversions** *angeführt.*

```
:.Z: : :/bin/compress -d -c %s:T_REG|T_ASCII:O_UNCOMPRESS:UNCOMPRESS
: : :.Z:/bin/compress -c %s:T_REG|T_ASCII:O_COMPRESS:COMPRESS
:.gz: : :/bin/gzip -cd %s:T_REG|T_ASCII:O_UNCOMPRESS:GUNZIP
: : :.gz:/bin/gzip -9 -c %s:T_REG|T_ASCII:O_COMPRESS:GZIP
: : :.tar:/bin/tar -c -f - %s:T_REG|T_DIR:O_TAR:TAR
: : :.tar.Z:/bin/tar -c -Z -f - %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+COMPRESS
: : :.tar.gz:/bin/tar -c -z -f - %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+GZIP
```

*Die interessantesten sind die beiden letzten Zeilen. Gibt es auf einem FTP-*Server *z.B. ein Verzeichnis* **latex**, *so kann durch*

[24] eine Datei mit Voreinstellungen liegt der Distribution bei

`ftp>` **get latex.tar.gz**

das komplette Verzeichnis **latex***, archiviert und komprimiert, geholt (mit* **put** *geschickt) werden.*

2.2.2 Spiegelung von FTP-Archiven

Die bedeutendsten FTP-*Server* haben in der Regel zwei Nachteile — sie liegen weit weg (meistens in den USA) und sie sind häufig überlastet. Aus diesen Gründen haben viele europäische FTP-*Server* das Spiegeln (*mirrowing*) interessanter Verzeichnisse eingeführt. Das Spiegeln besteht darin, die ausgewählten Pakete von den FTP-*Servern* automatisch zu Zeiten der geringsten Belastung herunterzuladen und dadurch einen lokalen FTP-*Mirror* zu schaffen.

Programmtechnisch wird das Spiegeln häufig mit Hilfe von Skript-Sprachen bewerkstelligt. Für größere Archive hat sich dabei das Programm **mirror** am besten bewährt. Es ist in der Sprache Perl geschrieben und erledigt automatisch alle notwendigen Aufgaben — Kontaktaufnahme zum *Server*, Vergleich des eigenen mit dem Inhalt des FTP-*Servers* und Anpassung des FTP-*Mirrors* an den aktuellen Stand.

Eine andere interessante Alternative bietet die objektorientierte Programmiersprache Python, in deren Lieferumfang das Modul **ftplib** gehört. Dieses definiert die Klasse FTP, die zur Implementierung des FTP-Protokolls auf seiten des *Clients* verwendet werden kann. Die damit geschriebenen Skripte ermöglichen die Automatisierung unterschiedlicher Aufgaben von FTP, zu denen z.B. auch das Spiegeln von anderen FTP-*Servern* gehört.

2.3 WWW — World Wide Web

Ende der achtziger Anfang der neunziger Jahre sind kurz nacheinander zwei neue Informationsdienste im *Internet* entstanden. Beide verfolgten das gleiche Ziel — Informationen den Benutzern leichter zugänglich zu machen. Der erste solche Dienst war Gopher[25]. Der Benutzer verbindet sich mittels Gopher-*Client* mit dem *Server* auf *Port* 70 und erhält daraufhin von ihm ein Menü mit Punkten zur Auswahl. Wählt er einen Menüpunkt aus, so erhält er ein neues Menü oder eine Datei, die automatisch auf seinen Rechner übertragen wird. Sämtliche Bedienungsschritte erfolgen über die Tasten `Enter` und `Esc`. Der Benutzer kann sich so von einem Menüpunkt zum anderen durch das gesamte *Internet* bewegen (*surfen*).

Gopher

Port 70

Das Gopher-Protokoll stellt eine Verbindung zwischen *Client* und *Server* nur zur Übertragung von Anfrage und Antwort her, danach wird die Verbindung wieder abgebrochen und am *Server* sämtliche Informationen über die stattgefundene Aktion gelöscht.

Jeder Menüeintrag enthält folgende Informationen:

typ name selektor server port

[25] engl. Erdeichhörnchen, ein Nager der Familie Geomyidae; Bezeichnung für die Einwohner Minnesotas (Gopher State); Laufbursche

Der *name* erscheint dem Benutzer im Menü. Die beiden folgenden Angaben, *server* und *port*, geben an, welcher Gopher-*Server* kontaktiert werden soll. Der *selektor* bestimmt, welche Daten von diesem *Server* übertragen werden, wenn der Benutzer den betreffenden Menüpunkt auswählt. Am Anfang jeder Zeile steht der Datentyp. Dieser bestimmt die Art der Information (→ Tab.2.5).

typ	Bedeutung
0	Textdatei
1	Verzeichnis (Menü)
2	PH (CSO) Telefonverzeichnis
4	**hqx**-Datei für Macintosh-Rechner
7	Suchen
8	Telnet
9	Binärdatei
s	Ton
h	WWW-Seite (HTML-Datei)

Tab. 2.5: Informationstypen beim Gopher

URL

Alle Eigenschaften des Gophers hat auch das WWW, jedoch in ausgereifterer Form. Die Angaben *server*, *port* und *selektor* wurden zur URL (*Uniform Resource Locator*), der *typ* wurde durch die MIME-Klassifizierung, das Menü durch ein Hypertext-Dokument, ersetzt.

HTML

Das Dokument ist in der Sprache HTML (*HyperText Markup Language*) geschrieben. Der *Client* hat die Aufgabe, das Dokument zu analysieren, zu formatieren und dem Benutzer darzustellen. Da mit HTML nur die logische Struktur des Dokuments, nicht jedoch das graphische oder typographische Aussehen bestimmt wird, hängt seine Darstellung vom verwendeten *Client*[26] ab.

HTTP

Zur Kommunikation zwischen *Client* und *Server* wird im WWW das Protokoll HTTP (*HyperText Transfer Protocol*) verwendet. Auch bei HTTP handelt es sich um ein Protokoll, das die Verbindung zwischen den beiden Rechnern nur zur Übertragung der Daten aufrechterhält. Der *Server* löscht danach alle Angaben zur stattgefundenen Aktion, eine erneute Anfrage kann daher auf die vorhergehende nicht Bezug nehmen.

Die wichtigste Komponente der Anfrage ist die URL. Mit ihrer Hilfe bestimmt der *Client*, welche Information er von welchem *Server* erhalten möchte.

Die Antwort des *Servers* beginnt mit einer Statuszeile, die den *Client* darüber informiert, ob die Anfrage erfolgreich war oder nicht. Der Rest ähnelt einem Brief der elektronischen Post.

Es folgen der Kopf, eine Leerzeile und der Rumpf — diesmal jedoch in 8-Bit-Kodierung, so daß beliebige (auch binäre) Daten übertragen werden können. Der Inhalt des Dokuments wird mit Hilfe des Schlüsselwortes `Content-Type` (→ S.46) spezifiziert.

Beispiel. *Für HTTP existiert ein spezieller Subtyp des Schlüsselwortes, nämlich*

```
Content-Type: text/html
```

[26] auch *Browser* genannt

So bezeichnete Dokumente sollten entweder in Standard-ASCII oder nach ISO 8859-1, kodiert sein. Soll eine andere Kodierung verwendet werden, so muß der entsprechende Zeichensatz explizit angegeben werden. Für mitteleuropäische Sprachen, die mit Hilfe von ISO 8859-2 kodiert werden, wäre die Anweisung wie folgt zu ergänzen:

```
Content-Type: text/html; charset=ISO-8859-2
```

Stehen dem Browser *die entsprechenden Zeichensätze zur Verfügung, so sollte er den Inhalt der Seite richtig darstellen können.*

Die Informationen liegen auf dem *Server* in der Regel in statischer Form vor. Die Seiten können jedoch auch dynamische Komponenten enthalten.

Interessante Möglichkeiten zum Entwurf dynamischer Seiten bietet die objektorientierte Programmiersprache Java. Ein Programm in dieser Sprache[27] kann in die HTML-Seite mit Hilfe spezieller *Tags*[28] eingefügt werden. Es wird dann zusammen mit der Seite übertragen und auf dem *Client* ausgeführt. Java

Dies hat einige praktische Konsequenzen:

- Der *Browser* muß das *Applet* ausführen (interpretieren) können.
- Während dem Ablauf des Programms ist keine Kommunikation zwischen *Client* und *Server* notwendig, die Geschwindigkeit, mit der das Programm abläuft, hängt daher nicht von der Qualität der Verbindung ab.
- Bei dieser Betriebsweise ist es momentan nicht möglich, z.B. die Identität des *Clients* festzustellen oder den Zugriff auf externe Datenbasen zu steuern, was ein gewisses Sicherheitsproblem darstellt.

Der *Client* kann Programme auch auf dem *Server* direkt starten. Diesem Zweck dient ein Satz von Vereinbarungen und Mechanismen, die als CGI (*Common Gateway Interface*) bezeichnet werden. Die Methode ist an keine konkrete Programmiersprache gebunden. Das CGI-Programm kann in C, als UNIX-Shellskript, oder in den höheren Skriptsprachen Perl, Python oder TCL geschrieben werden. Die Übergabe der Parameter[29] an das CGI-Programm erfolgt entweder über die Standardeingabe oder über die spezielle Shell-Variable[30] `QUERY_STRING`. Die Ergebnisse des CGI-Programms werden dem WWW-*Server* über die Standardausgabe übergeben. Damit ist es möglich, dem *Client* ein komplettes Dokument oder eine URL auf ein anderes Dokument[31] zu schicken. CGI

2.3.1 Adressierung im WWW

Als offizielle Adresse zur Identifikation eines bestimmten Ortes im WWW wurde das URI (*Uniform Resource Identifier*) eingeführt. Seine allgemeine Form ist

schema: *spezifikation-des-schemas*

[27] genannt, warum auch immer, *Applet*
[28] spezielle Marken, die eine neue logische Umgebung einleiten bzw. beenden
[29] z.B. der Inhalt von Formularen
[30] dies entspricht der Übergabe der Parameter beim Aufruf des Programms
[31] URL-*Redirection*

Die meisten *schemen* benutzen zur Identifikation der Informationen vorhandene *Internet*-Protokolle. Für ein URI mit diesen Eigenschaften wurde die Bezeichnung URL, für ein anderes, das keine konkreten Zugriffsmethoden voraussetzt, URN (*Uniform Resource Name*), eingeführt.

Die neuere RFC 1737 stellt ein anderes Modell vor. Der Begriff URI kommt darin praktisch nicht mehr vor. Zur Identifikation dient hier das URN. Dieses enthält keine Angaben über den Ort oder die Zugriffsmöglichkeit auf ein Objekt. Informationen z.B. über seine Kodierung, seinen Besitzer und die Zugriffsrechte stehen in den URC (*Uniform Resource Characteristics*), der Ort wird durch die URL angegeben.

Tatsache ist, daß der einzige feste Punkt im UR-Weltraum die URL ist. Nur diese ist klar definiert und wird intensiv genutzt. Alle anderen Begriffe sind Gegenstand der Diskussion.

Die Grundform der URL ist folgende:

schema`:`*spezifikation*

Mit *schema* wird die Methode angegeben, die den Zugriff auf bestimmte Informationen gewährleisten soll. Die folgende Tabelle (→ Tab.2.6) faßt die gängigsten Schemen zusammen. Die Angaben sind sicher vorläufig (wie so vieles zum Thema WWW und *Internet*). Mit Erweiterungen, z.B. `fax` für Telefax- oder `phone` für Telefon-Dienste, ist zu rechnen.

schema	Bezeichnung
`ftp`	*File Transfer Protocol*
`http`	*HyperText Transfer Protocol* (WWW)
`gopher`	Gopher-Protokoll
`mailto`	*e-mail*-Adresse
`news`	*Network News*
`telnet`	Telnet-Verbindung
`file`	Datei

Tab. 2.6: URL-Schemen

Die *spezifikation* hängt von dem benutzten Schema ab. In der Regel enthält sie den Namen des *Servers*, den Zugriffsweg, den Benutzernamen und ähnliches.

Obwohl die genaue Form in *spezifikation* von dem verwendeten Schema abhängt, ist der grundsätzliche Aufbau für die meisten gleich[32]:

`//`*benutzername*`:`*passwort*`@`*rechnername*`:`*port*`/`*weg*

benutzername Benutzername, notwendig z.B. für `telnet`.

passwort Passwort.

rechnername Name bzw. IP-Adresse des *Servers*.

port *Port* des gewählten Schemas. Die Angabe macht nur dann Sinn, wenn kein Standarddienst (→ S.4) angefordert wird. Der Standarddienst `http` hat z.B. die *Port*-Nummer 80, die zum Ansprechen des *Servers* implizit verwendet wird und daher nicht gesondert angegeben werden muß.

[32] z.B. `ftp://leibner:bl-b@ftp.xoom.com/index.html`

weg Den Rest der URL kann der Weg zur gewünschten Information bilden. Dabei ist zu beachten, daß der erste Schrägstrich nur zur Trennung des Weges vom Rest der URL dient. Er ist nicht Teil des Weges!

Als Teil des Weges können am Ende die Spezifikationen von Parametern und Anfragen auftreten. Um sie von dem eingentlichen Weg zu unterscheiden, werden sie durch einen Strichpunkt (Parameter) bzw. ein Fragezeichen (Anfragen) eingeleitet.

Um auf einen bestimmten Ort im aktuellen Dokument hinzuweisen, wird

`#`*abschnitt*

am Ende des Weges angegeben.

In der URL können alle geläufigen Zeichen verwendet werden — Buchstaben, Ziffern, Punkt, Bindestrich und der Unterstrich. Zugelassen sind auch alle anderen Zeichen aus dem unteren Teil der ASCII-Tabelle (32-126). Es wird jedoch empfohlen, sie zu kodieren (→ Tab.2.7).

Zeichen	Hex	Zeichen	Hex	Zeichen	Hex
TAB	09	Leerzeichen	20	"	22
%	25	+	2B	/	2F
:	3A	<	3C	>	3E
[	5B	\	5C	]	5D
^	5E	‘	60	{	7B
|	7C	}	7D	~	7E

Tab. 2.7: Verwendung von Sonderzeichen in der URL (Auswahl)

Soll z.B. `>` als Teil des Weges verwendet werden, so erfolgt die Angabe durch ein % gefolgt von zwei Hexadezimalziffern gemäß seiner Position innerhalb der ASCII-Tabelle, d.h. durch `%3E`.

Eine Ausnahme bildet das Leerzeichen. Dieses wird entweder durch `%20` oder ein `+` ersetzt.

Beispiel. *Soll z.B. der Text „*`dies ist: hokus+pokus`*“ als Teil der URL verwendet werden, so muß er wie folgt kodiert werden:*

```
dies+ist%3A+hokus%2Bpokus
```

HTTP

Wie bereits erwähnt, ist für HTTP der *Port* 80 reserviert. *Port* 80

Die URL für das HTTP-Protokoll sieht im allgemeinen wie folgt aus:

`http://`*rechnername*`:`*port*`/`*weg*`?`*suchtext*

Dabei ist der *rechnername* der Name oder die IP-Adresse des *Servers* und der *port* korrespondiert mit dem zuständigen Prozeß[33] der Applikation.

[33] z.B. dem *Daemon* **httpd**

Der *weg* führt zum ausgewählten Dokument auf dem *Server*. Fehlt er, so erhält der *Client* in der Regel die Heimatseite (*Homepage*) des *Servers*. Fehlt der Dateiname, so endet der Weg mit einem `/`.

Der *suchtext* war anfangs tatsächlich zum Suchen gedacht. Er wird heute im allgemeinen im Zusammenhang mit CGI-Programmen verwendet. Der *suchtext* dient dann zur Übergabe der Parameter an das CGI-Programm, falls die Methode `GET` verwendet wird.

Beispiel. *Alta Vista bietet unter der Adresse*

`http://www.altavista.com/`

ein Suchwerkzeug für das Internet *an. Nach Eingabe des Suchbegriffs* `phone.gif` *schickt der* Client *folgende URL an den* Server*:*

`http://www.altavista.com/cgi-bin/query?pg=q&what=web&fmt=.&q=phone.gif`

Der Suffix für WWW-Dokumente ist in der Regel **.html**. Arbeitet der *Server* in einer MS-Windows *xx* Umgebung, so lautet er **.htm**. Ein weiterer üblicher Suffix ist **.shtml**. Er wird für Seiten verwendet, die vom *Server* unmittelbar vor dem Abschicken durch das Einbinden von HTML-Texten aktualisiert werden. Dadurch können Dokumente erzeugt werden, die sich dynamisch ändern.

FTP

Das `ftp`-Schema wird für Dateien und Verzeichnisse verwendet, die mit dem Standarddienst FTP übertragen bzw. zugänglich gemacht werden sollen (*Port* 20 und 21). Zum Aufbau der Verbindung wird der Benutzername und sein Passwort verwendet. Fehlen diese Angaben, so erfolgt die Verbindung des *Client* zum *Server* nach den Regeln des anonymen FTP. Dem *Server* wird dann der Benutzername **anonymous** und als Passwort seine *e-mail*-Adresse übergeben.

Für *weg* kann folgendes stehen:

verz1`/`$\cdots$`/`*verzN*`/`*name*`;type=`*typ*

Die Aktion des *Clients* hängt vom *typ* ab. Zuerst geht der *Client* mittels **cd** an die angegebene Stelle auf dem *Server*, dann läßt er sich beim *typ*

`d` den Inhalt des Verzeichnisses *name* ausgeben, bei

`a` schaltet er in den Übertragungsmodus „**type ascii**“ und überträgt die ASCII-Datei *name* und bei

`i` wechselt er nach „**type binary**“ und überträgt die Binärdatei *name*.

Die Angabe des *typs* ist optional und fehlt bei den meisten URLs. Der *Client* muß dann erraten (z.B. anhand des Suffix), welche Art der Übertragung einzustellen ist.

Beispiel. *Wird dem* Client

`ftp://leibner@ftp.firma.de/pub/etwas.gz`

eingegeben, so verbindet er sich mit dem Server `ftp.firma.de` *unter dem Benutzernamen* `leibner`. *Benötigt der FTP-*Server *ein Passwort, so fragt der* Client *den Benutzer danach. Wird die Verbindung hergestellt, so wechselt der* Client *in das Verzeichnis* **pub** *und überträgt die Datei* **etwas.gz**.

Der Weg ist relativ angegeben. Im Beispiel wird es sich um ein Unterverzeichnis des Heimatverzeichnisses des Benutzers `leibner` *handeln.*

Soll die Angabe absolut erfolgen, so muß sie mit einem `/` *beginnen. Damit der* Client `/` *nicht als Teil des relativen Weges interpretiert, muß* `/` *kodiert werden. Da der ASCII-Kode des Schrägstriches* `2F` *ist, müßte die URL zur Angabe des absoluten Pfades*

`/pub/etwas.gz`

wie folgt aussehen:

`ftp://leibner@pd.siemens.de/%2Fpub/etwas.gz`

Mailto

Eines der einfachsten Schemen ist `mailto`. Es dient zur Übergabe von *e-mail*-Adressen und hat die folgende Form:

`mailto:`*e-mail-adresse*

Das in den *e-mail*-Adressen manchmal vorkommende %-Zeichen (→ S.42) muß in der URL durch `%25` ersetzt werden.

Bei der Verwendung von *Mailto* wird nur ein bestimmter Benutzer oder ein Programm angesprochen, die Verbindung ist daher nicht interaktiv.

Beispiel. *Die URL für die* e-mail-*Adresse eines Benutzers im* Internet *kann wie folgt aussehen:*

`mailto:Peter.Leibner@firma.de`

Ist der Adressat in einem anderen Netz beheimatet und ist die Adresse des in dieses Netz führenden Gateway-*Rechners bekannt, so kann die URL wie folgt geschrieben werden:*

`mailto:mueller%25csearn.bitnet@earn.cvut.cz`

Der Gateway-*Rechner* `earn.cvut.cz` *setzt dann die Internetadresse in die Adresse für das BITNET (→ S.42) um.*

News

Mit dem Schema *News* kann auf *News-Server* zugegriffen werden. Diese bieten „Network News“ (*Usenet News*) an. Die Aktualisierung und Auswahl der Gruppen (*Newsgroups*) wird durch den Administrator des *News-Servers* festgelegt.

Die Angabe des Namens einer Gruppe erfolgt hierarchisch. Einzelne Hierarchiestufen werden dabei durch einen Punkt unterteilt, z.B. `cz.comp.linux`.

Da durch Verweise im Schema `news` nicht nur Gruppen sondern auch konkrete Beiträge ausgewählt werden können, gibt es zwei unterschiedliche Darstellungen für die URL:

`news:`*newsgruppe*

`news:`*beitrags-identifikation*

Die *newsgruppe* bestimmt den Namen der Gruppe auf dem *News-Server*.

Die *beitrags-identifikation* (*Message-ID*) hat die Form

eindeutige-zeichenkette`@`*domaene*

Die *eindeutige-zeichenkette* wird durch den *News-Server* mit dem Namen *domaene* vergeben.

Beispiel. *Durch die Eingabe der URL*

`news:alt.binaries.pictures.girlfriends`

wird vom gewählten News-Server *(dieser muß entweder in einer Umgebungsvariablen oder z.B. bei Netscape unter* `Options: Mail & News Preferences` *spezifiziert werden) die Gruppe* `alt.binaries.pictures.girlfriends` *geladen und damit die Beiträge dieser Gruppe verfügbar. Wählt man z.B.* `Amateurs1.jpg` *aus, so stellt der* Browser *ein Bild im JPEG*[34]*-Format dar.*

Das Bild kann z.B. die beitrags-identifikation

`32F3C392.3ADA@ultranet.ca`

haben. Möchte man diesen Beitrag direkt ansprechen, so muß für die URL

`news:32F3C392.3ADA@ultranet.ca`

angegeben werden. Da Artikel von News-Servern *nur für eine begrenzte Zeit gespeichert werden, sollte die zweite Form in HTML-Dokumenten nicht verwendet werden.*

Das Schema enthält selbst keine Information über einen konkreten *News-Server*. Die URL `news:comp.os.linux` kann daher zwei Benutzern, die unterschiedliche *News-Server* benutzen, unterschiedliche Beiträge liefern.

Neben `news` existiert ein weiteres Schema zur Auswahl von „Network News“. Dieses Schema bietet ebenfalls Zugriffsmöglichkeiten auf *News-Server*, diesmal kann jedoch der Name eines konkreten *Servers* mit angegeben werden:

NNTP `nntp://`*rechnername*`:`*port*`/`*newsgruppe*

Die Verbindung erfolgt ebenso wie bei `news` mittels NNTP[35]. Der implizite Wert für den
Port 119 *Port* ist 119, die Angabe zur *newsgruppe* folgt den gleichen Konventionen wie bei `news`.

Obwohl durch `nntp` ein konkreter Beitrag im *Internet* ausgewählt werden kann, verbieten die meisten *News-Server* den Zugriff externer Benutzer auf ihre Daten. Der Dienst wird in der Regel nur Benutzern eines lokalen Netzes zur Verfügung gestellt (der *Browser* von Netscape versteht nur `news`, dieses Schema kann dann sowohl in der ersten Form als auch in der eben vorgestellten verwendet werden).

File

Das Schema `file` ist nicht zur Identifikation von Dateien im *Internet* gedacht. Mit seiner Hilfe kann z.B. auf Dateien zugegriffen werden, die sich auf dem lokalen Rechner befinden. Führt das Schema auf einen anderen Rechner, so wird es implizit durch `ftp` ersetzt.

Die grundlegende Form der URL ist folgende:

`file://`*rechnername*`/`*weg*

[34] *Joint Photographic Experts Group*, Format für die Kodierung von Photos
[35] *Network News Transfer Protocol*

Dabei steht *rechnername* für den voll qualifizierten Namen des Rechners, auf dem der *weg* zugänglich ist.

Der *weg* gibt den Pfad durch die Verzeichnisse an:

verz1/···/*verzN*/*dateiname*

Einen Spezialfall für *rechnername* stellt `localhost` oder eine leere Zeichenkette dar. In beiden Fällen wird dies als „der Rechner, an dem diese URL interpretiert wird“, verstanden.

Der im Schema `file` verwendete *weg* wird absolut angegeben.

Das Schema kann zum Auffinden einer Datei im lokalen Dateisystem benutzt werden. Dazu fängt man z.B. mit `file:///` oder `file://localhost/` im Wurzelverzeichnis des Dateisystems an und folgt durch Anklicken der jeweiligen Verzeichnisse dem Pfad bis an die Stelle, an der sich die Datei *dateiname* befindet.

2.3.1.1 Relative URLs

Alle bisher vorgestellten Schemen wurden durch absolute URLs angegeben. Sie enthielten vollständige Angaben darüber, wie und wo die gewünschten Informationen geholt werden sollen.

Wurde das Stammdokument erst einmal mittels absoluter URL geladen, so ist es nicht besonders sinnvoll, weitere Dokumente, die sich auf dieses beziehen und die zusammen mit ihm auf dem gleichen Rechner liegen, ebenfalls über absolute URLs anzusprechen. Vielmehr ist es einfacher und sicherer, den *weg* relativ zum Stammdokument anzugeben.

Zum Holen von Informationen muß jedoch immer der komplette Verweis bekannt sein — die absolute URL. Der *Client* setzt daher die Angaben, die er aus der Analyse der Stamm- und relativen URL gewonnen hat wieder zu einer absoluten Adresse zusammen. Die so ermittelte absolute URL wird dann für den Zugriff auf die Information auf dem *Server* verwendet.

Diese Vorgehensweise ist natürlich nicht für jedes Schema (z.B. `mailto`, `news` oder `telnet`) geeignet, sondern nur für Schemen (z.B. `file`, `ftp`, `http` und `nntp`), deren URLs die Struktur

schema`://`*ort*`/`*weg*`;`*parameter*`?`*anfrage*`#`*abschnitt*

einhalten.

ort Bestimmt den Ort, wo im *Internet* die gewünschte Information geholt werden soll. Er enthält Angaben über die IP-Adresse (Name) und gegebenenfalls die Nummer des *Ports*. Der *ort* kann zusätzlich Informationen über den Benutzernamen und sein Passwort enthalten (→ S.70).

weg Bestimmt die Stelle, an der sich die Informationen auf dem angesprochenen *Server* befinden.

parameter Sie enthalten ergänzende Informationen zum gewünschten Objekt (wie z.B. `type=a` beim Schema `ftp`).

anfrage Ermöglicht die Übergabe von Suchbegriffen an den *Server*.

abschnitt Bestimmt die Position innerhalb eines Dokumentes.

Ermittlung der Stamm-URL

Zur Bestimmung der Stamm-URL untersucht der *Client* die möglichen Alternativen in der nachfolgend gegebenen Reihenfolge.

1. Der *Client* untersucht zuerst, ob die Stamm-URL explizit durch den HTML-*Tag* `<BASE>` spezifiziert wird. Ist dies der Fall, so wird diese zur Stamm-URL.
2. Ist das Dokument Teil einer zusammengesetzten Menge von Dokumenten, z.B. der Form `multipart`, so wird die URL des übergeordneten Dokuments als Stamm-URL verwendet.
3. Treffen weder 1. noch 2. zu, so wird die URL verwendet, mittels derer das Dokument geholt wurde. Diese Regel kommt am häufigsten zur Anwendung.
4. Kann auch mit 3. aus irgendwelchen Gründen keine Stamm-URL ermittelt werden, so wird diese als leer angenommen und alle lokalen URLs gelten als absolut.

Analyse der relativen URL

Zur Analyse der relativen URL führt der *Client* folgende Schritte durch:

1. Er ermittelt die Stamm-URL, wie oben beschrieben.
2. Er zerlegt danach die Stamm- und relative URL in die auf Seite 75 angegebenen Teile der Struktur.
   a) Ist die analysierte URL leer, so wird sie auf den Wert der ermittelten Stamm-URL gesetzt und der Algorithmus beendet.
   b) Fängt die analysierte URL mit dem Namen eines Schemas an, wird sie als absolut angenommen und der Algorithmus ebenfalls beendet.
   c) Beginnt sie nicht mit der Angabe eines Schemas, so erbt sie dieses von der Stamm-URL und der Algorithmus wird mit dem folgenden Punkt fortgesetzt.
3. Ist die Angabe *ort* nicht leer, so wird mit Punkt 7. fortgefahren. Ist er leer, so erbt die analysierte URL diesen von der Stamm-URL.
4. Fängt der *weg* mit `/` an, so wird die analysierte URL nicht als relativ angenommen und der *Client* setzt den Algorithmus mit Punkt 7. fort.
5. Ist der *weg* der analysierten URL leer, so erbt sie diesen von der Stamm-URL.
   a) Sind die *parameter* der untersuchten URL nicht leer, so wird mit Schritt 7. fortgefahren.
   b) Sind sie leer, so erbt die analysierte URL diese von der Stamm-URL und der *Client* untersucht als nächstes die *anfrage*. Ist sie nicht leer, so setzt der *Client* den Algorithmus mit Punkt 7. fort. Ist sie leer, so erbt die untersuchte URL diese zuvor von der Stamm-URL und der Algorithmus geht dann zu Schritt 7. über.
6. Ist der *weg* nicht leer und fängt er nicht mit `/` an, erfolgt die Modifikation des *weges*. Der am weitesten rechts stehende Teil des *weges* der Stamm-URL wird durch den *weg* der analysierten URL ersetzt. Auf das Ergebnis werden danach folgende Schritte angewendet:

a) Überall, wo `./` steht, wird es aus der URL entfernt. Das gleiche geschieht mit einem eventuellen Punt am Ende des *weges*.

   Es werden dabei nur Punkte entfernt, die ein vollständiges Segment des *weges* sind. Dadurch wird verhindert, daß Punkte entfernt werden, die Teil eines Names sind.

b) Alle Vorkommen von *wegsegment*`/../` werden ausgewertet und entfernt (`..` führt einen Schritt zurück in der Hierarchie des *weges*).

   Bei *wegsegment*`/..` wird ebenso verfahren.

7. Die ermittelten Teile des Schemas werden zusammengesetzt, wodurch die absolute URL für den analysierten Verweis entsteht.

Die Zusammensetzung einer Stamm-URL mit unterschiedlichen relativen URLs soll anhand eines Beispiels erläutert werden.

Beispiel. *Als Stamm-URL sei*

`http://www.igw.de/html/doc/inhalt.html`

gegeben. In der nachfolgenden Tabelle sind die Ergebnisse verschiedener Kombinationen der relativen URLs mit der Stamm-URL wiedergegeben.

relative URL	Ergebnis
`kap1.html`	`http://www.igw.de/html/doc/kap1.html`
`../index.html`	`http://www.igw.de/html/index.html`
`#bild1`	`http://www.igw.de/html/doc/inhalt.html#bild1`
`/home`	`http://www.igw.de/home`
`ftp://ftp.igw.de/`	`ftp://ftp.igw.de/`
`..`	`http://www.igw.de/html/`
`../`	`http://www.igw.de/html/`
`../..`	`http://www.igw.de/`

Nun soll untersucht werden, wie die relative URL

`text.html`

unterschiedliche Stamm-URLs modifiziert.

Stamm-URL	Ergebnis
`http://www.igw.de/index.html`	`http://www.igw.de/text.html`
`http://www.igw.de/htm/index.html`	`http://www.igw.de/htm/text.html`
`ftp://ftp.igw.de/`	`ftp://ftp.igw.de/text.html`
`file:///doc/inhalt.html`	`file:///doc/text.html`

2.3.2 HTML in Beispielen

HTML ist von der *Standard Generalized Markup Language* (*SGML*[36]) abgeleitet. SGML wurde unter der Nummer 8879 als ISO-Norm festgeschrieben. SGML

[36] `http://www.sil.org/sgml/sgml.html`

Mit dem Begriff *Markup Language* werden Sprachen bezeichnet, deren Befehle zum Satz des Dokumentes Teil des Quelltextes sind. Es handelt sich dabei um eine Art von Programmiersprachen zur Gestaltung von Texten.

Zur Menge dieser Sprachen gehören neben SGML und HTML z.B. TEX (mit der Makroerweiterung LATEX, die z.B. zum Schreiben dieses Buches verwendet wurde) oder die UNIX-Familie ***roff** (**nroff**, **troff**, **groff**, usw.). Der wesentliche Unterschied zwischen den erstgenannten und TEX besteht darin, daß SGML und HTML nur die logische Struktur eines Dokumentes festhalten, während TEX Mittel und Wege definiert, die typographische Struktur eindeutig festzuschreiben. Dadurch wird ein in TEX gesetztes Dokument unabhängig vom Ausgabemedium in der Regel immer die gleiche Form haben, wogegen die Darstellung einer HTML-Seite je nach verwendetem *Browser* völlig unterschiedlich ausfallen kann.

Befehle (*Tags*) werden in HTML in spitze Klammern gesetzt, z.B. `<BASE>`. Zwischen Groß- und Kleinschreibung wird dabei nicht unterschieden. Es gibt *Tags*, die einzeln (z.B. `<IMG>`) oder paarweise (z.B. `<EM>` *text* `</EM>`) angegeben werden. Bei der paarweisen Angabe leitet der Beginn-*Tag* eine spezielle Umgebung ein, der Ende-*Tag* (mit `/` vor dem Befehlsnamen) beendet sie wieder.

Der allgemeine Aufbau einer Seite ist der folgende:

```
                <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>          <HTML>
                <HEAD>
<TITLE>         <TITLE>...</TITLE>
                   weitere Kopfzeilen
<HEAD>          </HEAD>
                <BODY>
                   HTML-Seite
                </BODY>
<BODY>          </HTML>
```

Obwohl HTML die eben angeführten *Tags* (bis auf `<TITLE>...</TITLE>`) nicht zwingend vorschreibt[37], wird dennoch empfohlen, sie zur Strukturierung des Dokumentes immer zu benutzen.

HTML ignoriert die Form des Quelltextes — das Zeilenende wird als Leerzeichen aufgefaßt, mehrere Leerzeichen hintereinander haben die gleiche Wirkung wie ein einziges. Das Zeilenende oder der Beginn eines neuen Absatzes muß daher explizit mit Hilfe von *Tags* angegeben werden (→ Tab.2.8).

`<P>`	neuer Absatz (*Paragraph*)
` `	neue Zeile (*line BReak*)
`<HR>`	waagrechter Strich (*Horizontal Rule*)

[37] Der komplette *Tag* `<TITLE>...</TITLE>` und die Definition `<!DOCTYPE...>` sind zwingend vorgeschrieben.

Anhand der *Document Type Definition* (DTD) wird die Menge der zulässigen SGML-Regeln — die Syntax der Sprache — definiert. Der *Browser* erkennt aus der DTD, um welche Version von HTML es sich handelt und wie er folglich die Seite darstellen soll. Die DTD sollte daher immer vorhanden sein.

Tab. 2.8: Strukturierung des Textes

<P> Jeder Absatz muß mit <P> beginnen und kann mit </P> beendet werden. Die Endemarke wird in der Regel jedoch nicht angegeben.

 Die Marke <BR> bewirkt einen Zeilenumbruch. Im Unterschied zu <P> wird jedoch kein zusätzlicher vertikaler Zwischenraum eingefügt. Um einen größeren vertikalen Zwischenraum zu erzeugen, ist

... einzugeben, eine bessere Lösung bietet HTML derzeit nicht an.

<HR> Zur Hervorhebung der Aufteilung kann zwischen den Absätzen ein horizontaler Strich eingefügt werden. Dazu dient die Marke <HR>. Sie wird in der Regel zusammen mit <P> verwendet: <P><HR><P>.

Viele *Tags* können durch Attribute modifiziert werden.

<*tag attribut*1 ... *attribut*n="*wert*n">

Es gibt Attribute mit oder ohne Angabe eines Wertes. Ist der *wert* eine Zahl oder eine vordefinierte Zeichenkette, so können die Anführungszeichen weggelassen werden.

ALIGN Ein für <P> interessantes Attribut ist ALIGN. Es kann die Werte LEFT (linksbündig, impliziter Wert), RIGHT und CENTER annehmen.

2.3.2.1 Schriften

Obwohl HTML eine Reihe von Befehlen zur Änderung der Schrift bereithält, ist das Ergebnis nicht eindeutig vorherzusagen. Es steht und fällt mit den Fähigkeiten des *Browsers*.

Aus diesem Grund wird empfohlen, logische (→ Tab.2.9) Hervorhebungen den absoluten (→ Tab.2.10) vorzuziehen.

<EM>	Hervorhebung (*EMphasize*)
<STRONG>	starke Hervorhebung (*STRONG emphasize*)
<CITE>	Zitat
<CODE>	Beispiel für Quellkode
<SAMP>	Beispiel für Zeichenkette (*SAMPle string*)
<KDB>	Tastatureingabe (*KeyBoarD input*)
<VAR>	Name einer Variablen (*VARiable*)

Tab. 2.9: Logische Hervorhebung

Alle *Tags* in den Tabellen 2.9 und 2.10 treten paarweise auf.

<B>	Fettschrift (*Bold*)
<I>	Kursivschrift (*Italics*)
<TT>	Nichtproportionalschrift (*TypewriTer*, *TeleType*)
<U>	unterschreichen (*Underscore*)
<BIG>	große Schrift
<SMALL>	kleine Schrift
<SUB>	tiefgestellter Index
<SUP>	hochgestellter Index

Tab. 2.10: Absolute Hervorhebung

Beispiel.

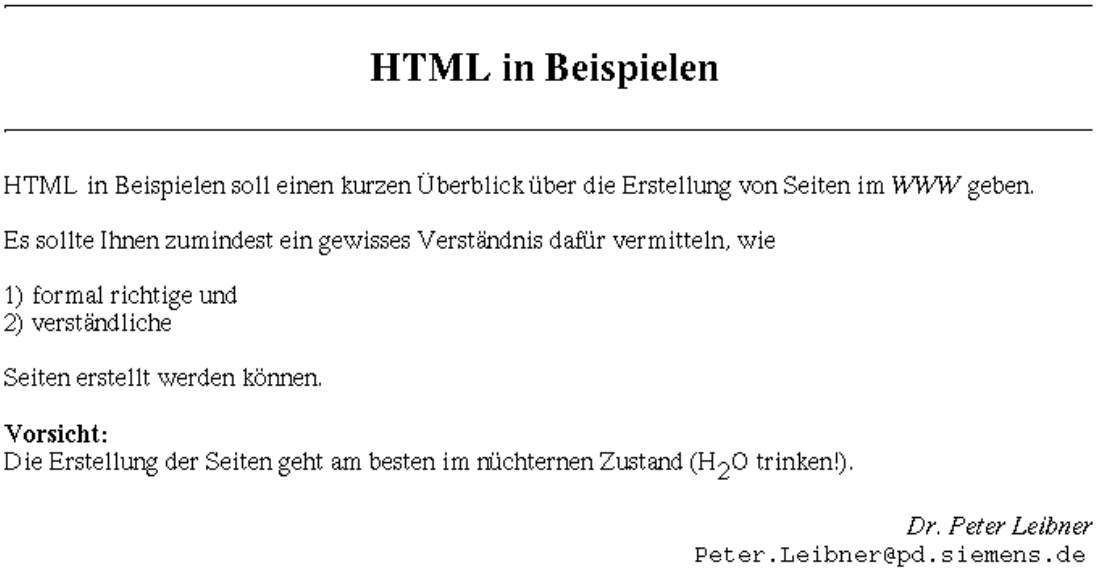
```
1   <HTML>
2   <HEAD>
3   <TITLE>HTML in Beispielen</TITLE>
4   </HEAD>
5   <BODY>
6
7   <P><HR><P ALIGN=CENTER>
8   <BIG><STRONG>HTML in Beispielen</STRONG></BIG>
9   <P><HR><P>
10
11  HTML in Beispielen soll einen kurzen &Uuml;berblick
12  &uuml;ber die Erstellung von Seiten im <EM>WWW</EM> geben.
13
14  <P>
15  Es sollte Ihnen zumindest ein gewisses Verst&auml;ndnis
16  daf&uuml;r vermitteln, wie<P>
17
18  1) formal richtige und<BR>
19  2) verst&auml;ndliche<P>
20
21  Seiten erstellt werden k&ouml;nnen.
22
23  <P>
24  <B>Vorsicht:</B><BR>
25  Die Erstellung der Seiten geht am besten im
26  n&uuml;chternen Zustand (H<SUB>2</SUB>O trinken!).
27
28  <P ALIGN=RIGHT>
29  <I>Dr. Peter Leibner</I><BR>
30  <TT>Peter.Leibner@pd.siemens.de</TT>
31
32  </BODY>
33  </HTML>
```

Ergebnis → Abb.2.4.

In Zeile 3 *ist der Titel des Dokuments angegeben. Diese Angabe sollte immer vorhanden sein, da sie der* Browser *erstens als Bezeichnung für seinen äußeren Rahmen und zweitens als Eintrag in die* Bookmarks *verwendet. Dabei sind besser keine HTML-Symbole (→ Tab.2.11) zu verwenden, da manche* Browser *diese falsch interpretieren.*

Die einzelnen Befehle sollten aus dem Vergleich der HTML-Datei mit der Ausgabe des Browsers *(→ Abb.2.4) klar sein.*

Im Hinblick auf das Internet *und seine internationale Anwendung, müssen für deutsche Zeichen (Umlaute und scharfes ß) die dafür vorgesehenen HTML-Sequenzen angegeben werden. Dies gilt auch für Sonderzeichen, die in HTML zur Markierung Verwendung finden (→ Tab.2.11).*



HTML in Beispielen

HTML in Beispielen soll einen kurzen Überblick über die Erstellung von Seiten im *WWW* geben.

Es sollte Ihnen zumindest ein gewisses Verständnis dafür vermitteln, wie

1) formal richtige und
2) verständliche

Seiten erstellt werden können.

Vorsicht:
Die Erstellung der Seiten geht am besten im nüchternen Zustand (H_2O trinken!).

Dr. Peter Leibner
Peter.Leibner@pd.siemens.de

Abb. 2.4: Bildschirmausgabe, Bsp.2.10

`&auml;`	ä	`<`	<
`&Auml;`	Ä	`>`	>
`&ouml;`	ö	`&`	&
`&Ouml;`	Ö	`&quot;`	"
`&uuml;`	ü	` `	Leerzeichen
`&Uuml;`	Ü	`&szlig;`	ß

Tab. 2.11: Auswahl von HTML-Symbolen

Die im Beispiel verwendeten Sonderzeichen können auch in der Form `&#`*kode*`;` angegeben werden. Dabei steht für *kode* der dezimale Wert der Position des Zeichens innerhalb der ASCII-Tabelle (sollte wegen unterschiedlichen Belegungen nationaler ASCII-Tabellen nicht verwendet werden).

Beispiel. *Für* © *kann daher sowohl* `&copy;` *als auch* `&#169;` *angegeben werden.*

Eine spezielle Bedeutung kommt ` ` zu. Dieses Zeichen entspricht dem Leerzeichen, es darf an dieser Stelle jedoch kein Zeilenumbruch erfolgen. Möchte man mehrere Leerzeichen im Text unterbringen (was mit simplen Leerzeichen nicht möglich ist), so muß entsprechend `   `..., eingegeben werden.

Ein Kommentar sieht in HTML wie folgt aus:

```
<!-- Dies ist ein Kommentar (→ Bsp.2.3.2.3, Zeilen 17-19) -->
```

oder

```
<!-- Dies ist ein JavaScript-Kommentar, der Quelltext
zwischen <SCRIPT> und </SCRIPT> einschließt (→ S.126)
//-->
```

2.3.2.2 Bilder

Die geläufigsten Bildformate haben zwei grundsätzliche Varianten. Ein Bild wird mit TrueColor bezeichnet, wenn jeder Bildpunkt (Pixel) eine beliebige Farbe aus ca. 16 Millionen möglichen hat, wird das Bild durch eine Farbpalette ergänzt, so kann es maximal 256 unterschiedliche Farben enthalten.

In Bildern, die als TrueColor bezeichnet werden, wird der Bildpunkt durch die Intensität der drei Farben R (*Red*), G (*Green*) und B (*Blue*) bestimmt. Für jede Farbe wird sie durch ein Byte bestimmt. Auf diese Art und Weise ist es möglich, 2^{8+8+8} Farben zu unterscheiden — eben die oben angegebenen 16 Millionen.

Bei der Verwendung einer Palette wird die Farbe durch ein Byte bestimmt. Dieses enthält nicht die Farbe selbst, sondern die Position der Farbe innerhalb der Farbpalette. Diese ist dann Teil des Bildes.

Bilder im Format TrueColor werden am häufigsten im JPEG-Format (mit gleichnamigem Kompressionsverfahren komprimiert) abgelegt, wogegen für Bilder mit Palette typischerweise GIF (*Graphics Interchange Formats*) mit dem Kompressionsverfahren LZW[38] verwendet wird. Da GIF auf einem patentierten Algorithmus basiert, soll es in Zukunft durch das patentfreie Format PNG (*Portable Network Graphics*) ersetzt werden.

JPEG

GIF

Da das menschliche Auge nur näherungsweise 64 Schattierungen einer Farbe zu unterscheiden vermag[39], sind die bei TrueColor möglichen 256 Stufen für jede RGB-Farbe als Luxus zu betrachten. Die Senkung der Nuancen auf 32 bei Rot und Blau und auf 64 bei Grün, reduziert die Anzahl der notwendigen Bit zur Färbung eines Pixels auf $5 + 5 + 6 = 16$ Bit, also zwei Byte. Diese Technik wird bei dem Format HighColor angewendet, das häufig in dem Betriebssystem Windows Verwendung findet.

Das RGB-Farbmodell geht von einem Farbwürfel aus, dessen drei Achsen 256 Schattierungen für Rot, Grün und Blau erlauben. Der Würfel selbst enthält somit alle darstellbaren Farben, wobei Schwarz im Ursprung (0,0,0) und Weiß auf der gegenüberliegenden Ecke (255,255,255) liegt. Die übrigen Ecken bilden die Farben Rot, Grün, Blau, Türkis, Violett und Gelb. Ein Punkt in diesem Würfel repräsentiert dann die Farbe eines Pixels im Format TrueColor.

Bei fester Farbpalette wird jede der drei Achsen der Länge 256 gleichmäßig unterteilt und die Schnittmenge in der Palette abgelegt. Am häufigsten erfolgt die Unterteilung durch $R \times G \times B = 8 \times 8 \times 4$.

Der Vorteil fester Farbpaletten liegt in ihrer Einfachheit und dem einfachen Zugang zu Informationen über die nächstgelegenen Nachbarn. Diese werden zur Bestimmung von farblichen Schattierungen benötigt. Der Nachteil liegt in der Überflüssigkeit mancher Farben, die zwar in der Farbpalette enthalten sind, zur Bildung eines neuen Bildes jedoch oft nicht benötigt werden (das Bild enthält z.B. kein Grün).

Aufgrund der Einfachheit und der Notwendigkeit mehrere unterschiedliche Bilder gleichzeitig darzustellen, werden die festen Farbpaletten z.B. in den *Browsern* Netscape oder Explorer verwendet. Diese verwenden feste Paletten der Form $R \times G \times B = 6 \times 6 \times 6$.

[38] Lempel-Ziv Welch; Kompressionsalgorithmus, den auch das UNIX-Kommando **compress** anwendet
[39] bei Grün etwas mehr

Sie nutzen damit nicht die ganze mögliche Farbskala von 256 Farben pro Pixel aus ($6^3 = 216$ Farben). Dieses Herabsetzen der Menge der Farben erzwingt Windows, das die Farbpalette mit Systemfarben für den eigenen Bedarf auffüllt. Damit die Bilder durch den *Browser* richtig dargestellt werden können, müssen sie mit seiner Farbpalette hergestellt worden sein.

`<IMG>` Bilder werden in die Seiten mit Hilfe des *Tags* `<IMG>` eingebracht. Diese Marke tritt nur einzeln, nicht paarweise auf. Sämtliche Eigenschaften werden durch ihre Attribute bestimmt (→ Tab.2.12).

`SRC` Das wichtigste Attribut ist `SRC`. Es bestimmt den Ort, wo das Bild geholt werden soll.

`ALT` Bei nichtgraphikfähigen *Browsern*[40] muß dem Benutzer statt des Bildes ein alternativer Text angeboten werden. Dies geschieht mit Hilfe von `ALT`. Die Marke `<IMG>` sollte niemals ohne dieses Attribut verwendet werden[41].

`SRC="..."`	URL des Bildes
`ALT="..."`	alternativer Text für nichtgraphikfähige *Browser*
`ALIGN="..."`	Plazierung bezüglich des umgebenden Textes
`ISMAP`	Bild mit Verweis
`WIDTH="..."`	Bildbreite
`HEIGHT="..."`	Bildhöhe

Tab. 2.12: Attribute von `<IMG>`

Ein Bild wird grundsätzlich wie jedes andere Textzeichen behandelt. Soll es allein stehen, so muß es durch `<P>` oder `<BR>` vom restlichen Text abgeteilt werden. Kommt das Bild als Zeichen einer Zeile vor, so kann es bezüglich der restlichen Zeichen dieser `ALIGN` Zeile angeordnet werden. Dies geschieht mit dem Attribut `ALIGN`. Dieses kann die Werte `TOP` (Anpassung der oberen Ränder), `BOTTOM` (Anpassung der unteren Ränder, impliziter Wert) und `MIDDLE` annehmen. Bei `MIDDLE` wird die Mitte des Bildes gegen die Grundlinie der Zeile ausgerichtet, wodurch die Mitte des Bildes etwas unterhalb der optischen Mitte der Zeile erscheint.

Mit den Werten `LEFT` oder `RIGHT` wird das Bild aus der Zeile herausgenommen und links oder rechts von ihr am Rand der Seite plaziert. Der Text läuft dann um diese Bilder herum (→ Bsp.2.12). Um mit einem Text hinter den Bildern fortzufahren, kann das `CLEAR` Umlaufen durch `<BR>` mit dem Attribut `CLEAR=ALL` beendet werden. Weitere Werte von `CLEAR` sind `LEFT` oder `RIGHT`, wodurch die neue Zeile unter das zuletzt umlaufene linke oder rechte Bild gesetzt wird.

`WIDTH` Die Attribute `WIDTH` und `HEIGHT` teilen dem *Browser* die Maße des einzubindenden Bildes in Pixeln mit. Sind sie angegeben, so kann der *Client* schon bei der Übertragung des `HEIGHT` Textes den notwendigen Platz für die Bilder freihalten und die Seite bereits darstellen, ohne auf die Maßangaben aus den Bilddateien warten zu müssen. Es ist darauf zu achten, daß die Maßangaben aus den Bilddateien mit denen der Attribute übereinstimmen,

[40] z.B. Lynx
[41] eine Syntaxkontrolle, z.B. durch **weblint**, meldet fehlende `ALT`-Attribute bei Bildern

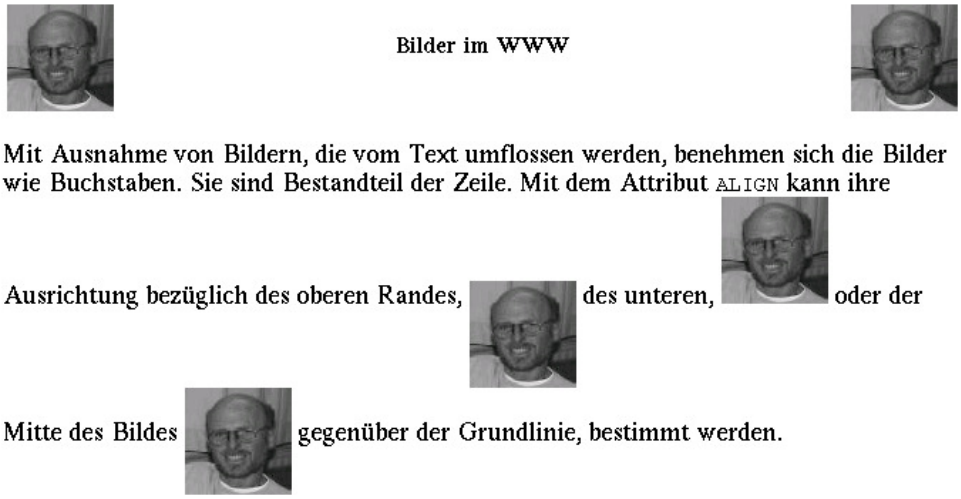


Abb. 2.5: Bildschirmausgabe, Bsp.2.12

da sonst der *Client* versucht, die Bilder nach den Werten der Attribute anzupassen. Dies muß nicht immer zum gewünschten Ergebnis führen.

Es ist sinnvoll, den Umfang der Bilddaten möglichst klein zu halten. Dazu dienen die oben erwähnten Kompressionsverfahren oder der Ersatz der Bilder durch ihre Verkleinerungen (*Icons*). Möchte der Benutzer das vollständige Bild in seiner ursprünglichen Größe sehen, so kann er das Bild durch Anklicken des *Icons* nachträglich vom *Server* holen.

Beispiel.

```
1   <P ALIGN=CENTER>
2   <IMG SRC="surfer_klein.gif" ALT="*" ALIGN=LEFT WIDTH=75 HEIGHT=75>
3   <IMG SRC="surfer_klein.gif" ALT="*" ALIGN=RIGHT WIDTH=75 HEIGHT=75>
4   <BR><BIG><STRONG>Bilder im WWW</STRONG></BIG><BR CLEAR=ALL>
5
6   <P>
7   Mit Ausnahme von Bildern, die vom Text umflossen werden,
8   benehmen sich die Bilder wie Buchstaben.
9   Sie sind Bestandteil der Zeile.
10  Mit dem Attribut <CODE>ALIGN</CODE> kann ihre Ausrichtung
11  bez&uuml;glich des oberen Randes,
12  <IMG SRC="surfer_klein.gif" ALT="*" ALIGN=TOP WIDTH=75 HEIGHT=75>
13  des unteren,
14
15  <IMG SRC="surfer_klein.gif" ALT="*" ALIGN=BOTTOM WIDTH=75 HEIGHT=75>
16  oder der Mitte des Bildes
17
18  <IMG SRC="surfer_klein.gif" ALT="*" ALIGN=MIDDLE WIDTH=75 HEIGHT=75>
19  gegen&uuml;ber der Grundlinie, bestimmt werden.
```

Ergebnis → Abb.2.5.

Die Zeilen 1–10 *und* 31–33 *aus dem Beispiel 2.10 sind hier und in den folgenden Beispielen, falls nicht anders angegeben, entsprechend hinzuzufügen.*

2.3.2.3 Verweise

<A> Ein Verweis wird in ein Dokument mit dem *Tag* <A> (*Anchor*) eingefügt. Er enthält zwei Informationen: Wohin er führt und wie er im *Browser* dargestellt werden soll. Das
HREF Ziel wird durch das Attribut HREF (*Hypertext Reference*) angegeben. Die Darstellung bestimmt der Text oder das Bild, die von <A> und </A> eingeschlossen werden.

Beispiel. *Erhält der* Browser *den Quelltext*

```
Die Schulseite der
<A HREF="http://www.siemens.de/schule/">Siemens Technik Akademie</A>
ist nur &uuml;ber das <EM>Internet</EM> zu erreichen.
```

so erhält der Benutzer durch Anklicken des Textes Siemens Technik Akademie *die* Homepage *der Schule. Der aktive Text wird in der Regel vom restlichen farblich und durch Unterschreichung hervorgehoben.*

Wird durch <A> ein Bild eingeschlossen, so erscheint es implizit mit einem umgebenden Rahmen. Da dies in der Regel nicht gut aussieht, erhielt die Marke <IMG> das Attribut
BORDER BORDER. Ihr Wert bestimmt die Dicke des Rahmens in Pixeln. Die Angabe BORDER=0 verhindert, daß ein Rahmen sichtbar wird.

Wie bereits erwähnt, kann mit Hilfe von <A> auch ein Verweis auf eine Stelle im aktuellen
NAME Dokument erfolgen. Dazu muß die Stelle durch das Attribut NAME markiert werden.

Beispiel.

```
<A NAME="instal"><STRONG>Installationshinweise</STRONG></A>
```

Der Verweis an diese Stelle erfolgt dann mit Hilfe der Angabe #*abschnitt* (→ S.71) im Attribut HREF.

Beispiel.

```
Weitere Angaben sind im Kapitel
<A HREF="#instal">Installationshinweise</A> zu finden.
```

Bei der Verwendung von Verweisen innerhalb eines Dokumentes ist zu beachten, daß der *Browser* den gesamten Text davor einlesen muß, um zu wissen, wie er die Stelle zu formatieren hat. Mit einem Verweis an das Ende einer langen Seite wird also nichts gewonnen, im Gegenteil, da der *Browser* den Text vor dem Anker zuerst lesen muß, ihn dabei aber nicht darstellt, erscheint der ganze Vorgang optisch verlangsamt.

Beispiel.

```
1   <BODY BGCOLOR="#FFFFFF">
2
3   <A HREF="#stae">Postanschrift Erlangen</A>
4
5   <P ALIGN=CENTER>
6   <A NAME="stam"><B>Postanschrift M&uuml;nchen</B></A>
7
8   <P ALIGN=CENTER>
9   STA M&uuml;nchen<BR>
10  Koppstra&szlig;e 6<BR>
11  81379 M&uuml;nchen<BR>
12  Tel +49 89 722 2 61 13<BR>
13  Fax +49 89 722 6 25 35<BR>
14
15  <HR WIDTH=340><P ALIGN=CENTER>
16
17  <!-- HR wirkt wie P
18  und wird implizit zentriert
19  -->
20
21  <B>E-Mail-Adresse</B><BR>
22  <P ALIGN=CENTER>
23  <A HREF="mailto:Peter.Leibner@pd.siemens.de">
24  Peter.Leibner@pd.siemens.de</A>
25
26  <HR width=340><P ALIGN=CENTER>
27  <B>Internet- und Intranet-Adressen der STA</B>
28  <P ALIGN=CENTER>
29  <A HREF="http://www.siemens.de/schule/">Internet</A><BR>
30  <A HREF="http://info.mchw.siemens.de/ukif/schule/">Intranet</A>
31
32  <P ALIGN=CENTER><IMG SRC="balonek.gif" HEIGHT=200 WIDTH=150><P>
33
34  <HR width=340><P ALIGN=CENTER>
35  <A NAME="stae"><B>Postanschrift Erlangen</B></A>
36  <P ALIGN=CENTER>
37
38  STA Erlangen<BR>
39  Zeppelinstra&szlig;e 10<BR>
40  91052 Erlangen<BR>
41  Tel +49 9131 74 61 58<BR>
42  Fax +49 9131 72 91 91<BR>
43
44  <P ALIGN=CENTER><IMG SRC="balonek.gif" HEIGHT=350 WIDTH=150><P>
45  <A HREF="#stam">M&uuml;nchner Adressen</A>
```

Ergebnisse → Abbn.2.6 und 2.7.

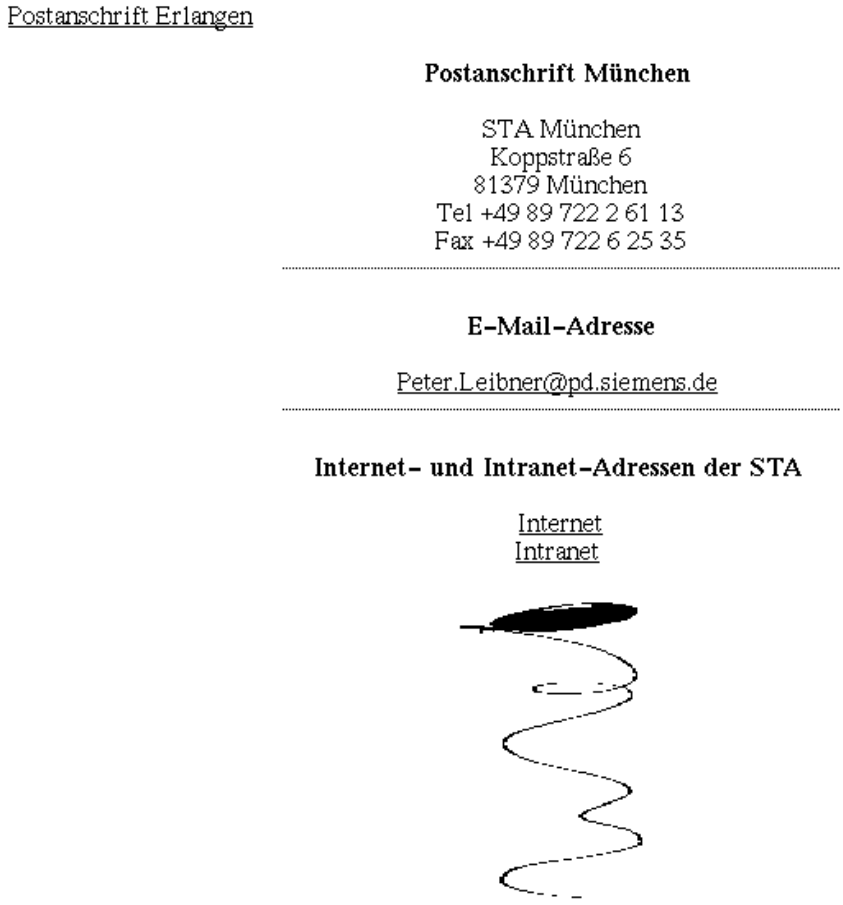


Abb. 2.6: Bildschirmausgabe, Bsp.2.3.2.3

In Zeile 1 *wurde durch* `BGCOLOR="#FFFFFF"` *die Hintergrundfarbe auf Weiß eingestellt. Erwähnenswert ist außerdem der Kommentar in den Zeilen* 17–19*. Der Rest sollte aus dem Vergleich mit den* Browser-*Ausgaben in den Abbildungen 2.6 und 2.7 klar sein.*

Wird die Angabe `Postanschrift Erlangen` *angeklickt, so erhält man die Ausgabe in Abbildung 2.7. Durch das Anklicken von* `Münchner Adressen` *kehrt man auf die Ausgabe in Abbildung 2.6 zurück.*

2.3.2.4 Formatieren der Seiten

Der Aufbau einer Seite gliedert sich normalerweise in Überschrift, Textteil und Schluß.

`<Hn>` Für Überschriften wird die Marke `<Hn>` verwendet. Für n können Ziffern von 1–6 stehen. `<H1>` schließt die Überschrift einer ganzen Seite ein — sie ist daher die größte. Jede Seite sollte nur eine Überschrift haben. Die einzelnen Teile der Seite werden durch Überschriften niedrigerer Kategorie (2–6) eingeleitet. Der paarweise auftretende *Tag* hat
`ALIGN` das Attribut `ALIGN`. Es kann die Werte `LEFT` (implizit), `RIGHT` und `CENTER` annehmen.

`<DIV>` Zur Strukturierung der Seite dient auch die Marke `<DIV>`. Mit ihrer Hilfe kann ein Teil
`ALIGN` des Textes logisch zusammengefaßt werden. Als Attribut ist `ALIGN`, mit den Werten

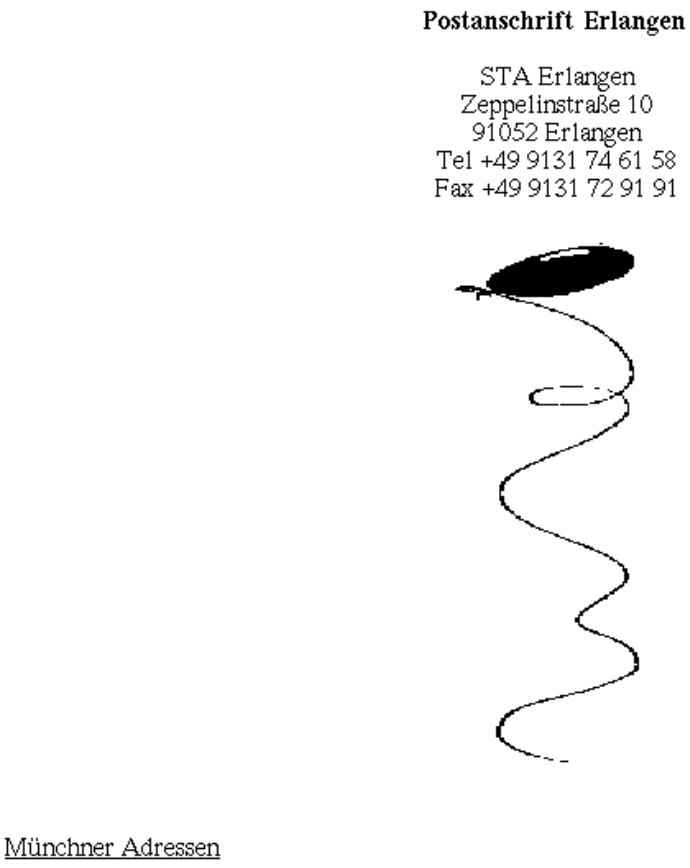


Abb. 2.7: Bildschirmausgabe, Bsp.2.3.2.3ff

`LEFT`, `RIGHT` und `CENTER`, zugelassen.

HTML bietet drei unterschiedliche Listenstrukturen an (→ Tab.2.13).

`<OL>`	durchnumerierte Liste (*Ordered List*)
`<UL>`	Aufzählungsliste (*Unordered List*)
`<LI>`	Listeneintrag in `<OL>` und `<UL>`
`<DL>`	Beschreibungsliste (*Definition List*)
`<DT>`	Begriff (*Definition Term*)
`<DD>`	Beschreibung (*Definition Description*)

Tab. 2.13: Marken zur Definition von Listenstrukturen

Die Marken `<OL>` und `<UL>` treten paarweise auf. Sie unterscheiden sich nur in der Hervorhebung der einzelnen Punkte der Liste.

Bei `<OL>` werden die durch `<LI>` eingeleiteten Punkte durchnumeriert. Das Attribut `TYPE` kann die in Tabelle 2.14 angegebenen Werte annehmen. `<OL>` `TYPE`

Für die Numerierung kann mit dem Attribut `START` ein Anfangswert vorgegeben werden. Es handelt sich dabei immer um einen numerischen Wert, auch wenn eine andere Art der Aufzählung gewählt wurde. In jedem weiteren Punkt der Liste wird dieser Wert `START`

VALUE um eins erhöht, außer dem *Tag* <LI> wird durch das Attribut VALUE ein anderer Wert zugewiesen.

<UL> Die Marke <UL> kennzeichnet die duch <LI> bestimmten Punkte je nach Schachtelungstiefe[42] durch ein für die jeweilige Ebene reserviertes Symbol. Diese implizite Einstellung
TYPE kann durch TYPE geändert werden. Die möglichen Werte für TYPE sind DISC (impliziter erster Wert), CIRCLE (impliziter zweiter Wert) und SQUARE (impliziter dritter Wert).

Wert	Bedeutung
1	arabische Ziffern
i	kleine römische Ziffern
I	große römische Ziffern
a	kleine Buchstaben
A	große Buchstaben

Tab. 2.14: Werte des Attributs TYPE der Marke <OL>

<DL> Die Begriffsliste wird von den *Tags* <DL> und </DL> umschlossen. Jeder Begriff wird
<DT> durch <DT> definiert. Dabei bleibt die Schrift unbeeinflußt. Soll er hervorgehoben werden, so muß dies explizit, z.B. durch die Marke <STRONG>, geschehen. Jedem Begriff folgt seine Erklärung. Diese wird nach rechts eingerückt dargestellt und durch die Marke
<DD> <DD> eingeleitet.

<BLOCKQOUTE> Zum beidseitigen Einrücken eines Textes dient die Marke <BLOCKQUOTE>.

<ADDRESS> Für Postadressen wurde in HTML der *Tag* <ADDRESS> eingeführt, der ebenso wie der oben angeführte *Tag* <BLOCKQUOTE> paarweise auftreten muß. Der *Tag* hat vorwiegend logische Bedeutung. Man hofft, daß er einmal von automatischen Suchwerkzeugen bei der Erstellung von Datenbanken verwendet wird.

<PRE> Vorformatierter Text kann mit der paarweise auftretenden Marke <PRE> erreicht werden. Dadurch werden die automatischen Formatierungsmechanismen unterdrückt und der Text etsprechend der Eingabe auf dem Bildschirm dargestellt. Es werden alle Leerzeichen und Zeilenumbrücke berücksichtigt, der Text wird durch eine Nichtproportionalschrift ausgegeben.

Verboten ist es, innerhalb von <PRE> Formatierungsbefehle (Überschriften, Listen und ähnliches) zu verwenden, erlaubt ist es dagegegen, Marken zur Änderung der Schrift (in der Regel bleibt die nichtproportionale Schrift erhalten, sie kann jedoch geneigt oder fett erscheinen) und Verweise, zu verwenden.

Beispiel.

```
1   <H1>Das Textsatzsystem LaTeX</H1>
2   <H2>TeX und LaTeX</H2>
3
4   <OL>
5   <LI>TeX
6
```

[42] es werden nur drei Ebenen unterschieden

```
7   <UL>
8   <LI>entwickelt von <EM>Donald E. Knuth</EM>
9   <LI>zum Setzen und Drucken von Texten und
10  mathematischen Formeln
11  <LI><STRONG>Wie</STRONG> soll gesetzt werden?
12  <UL>
13  <LI>Formatierungsdirektiven
14  </UL>
15  <LI>Buch-<STRONG>Setzer</STRONG>
16  </UL>
17
18  <LI>LaTeX
19
20  <UL>
21  <LI>Makropaket von <EM>Leslie Lamport</EM>, setzt
22  auf TeX auf
23  <LI>Textformatierung anhand vorgefertigter Layouts
24  <LI><STRONG>Was</STRONG> soll gedruckt werden?
25  <UL>
26  <LI>Strukturierungsdirektiven
27  </UL>
28  <LI>Buch-<STRONG>Designer</STRONG>
29  </UL>
30  </OL>
31
32  <H3>Beispiel</H3>
33  <BLOCKQUOTE><PRE>
34  Der Koeffizient $a_{-1}$ wird \textbf{Residuum}
35  von $w(z)$ an der Stelle $z = \alpha$ genannt.
36  \[
37  \textnormal{Res}(w,\alpha):= a_{-1}=\frac{1}{j2\pi}
38  \oint\limits_c w(z)\,dz
39  \]
40  </PRE></BLOCKQUOTE>
41
42  <H4>Auswahl an Literatur</H4>
43  <DL>
44  <DT><STRONG>Goosens, Mittelbach, Samarin</STRONG>
45  <DD>Der LaTeX-Begleiter, Addison-Wesley (Deutschland) GmbH<BR>
46  1995, ISBN 3-89319-646-3
47  <DT><STRONG>Kopka</STRONG>
48  <DD>LaTeX-Einf&uuml;hrung, Band 1, Addison-Wesley
49  (Deutschland) GmbH<BR> 1994, ISBN 3-89319-664-1
50  </DL>
```

Ergebnis → Abb.2.8

Wie man an der Ausgabe der Seite leicht erkennen kann, stellt Netscape in der Version 3.04 die Überschriften nicht korrekt dar. Mit solchen Überraschungen ist leider immer

Das Textsatzsystem LaTeX

TeX und LaTeX

1. TeX
   - entwickelt von *Donald E. Knuth*
   - zum Setzen und Drucken von Texten und mathematischen Formeln
   - **Wie** soll gesetzt werden?
     - Formatierungsdirektiven
   - Buch–**Setzer**
2. LaTeX
   - Makropaket von *Leslie Lamport*, setzt auf TeX auf
   - Textformatierung anhand vorgefertigter Layouts
   - **Was** soll gedruckt werden?
     - Strukturierungsdirektiven
   - Buch–**Designer**

Beispiel

```
Der Koeffizient $a_{-1}$ wird \textbf{Residuum}
von $w(z)$ an der Stelle $z = \alpha$ genannt.
\[
\textnormal{Res}(w,\alpha):= a_{-1}=\frac{1}{j2\pi} \oint\limits_c w(z)
\]
```

Auswahl an Literatur

Goosens, Mittelbach, Samarin
Der LaTeX–Begleiter, Addison–Wesley (Deutschland) GmbH
1995, ISBN 3–89319–646–3

Kopka
LaTeX–Einführung, Band 1, Addison–Wesley (Deutschland) GmbH
1994, ISBN 3–89319–664–1

Abb. 2.8: Bildschirmausgabe, Bsp.2.14

zu rechnen.

Die in den Zeilen 36–39 *angegebene Funktion ergibt nach der Übersetzung mit LaTeX folgendes:*

$$\mathrm{Res}(w,\alpha) := a_{-1} = \frac{1}{j2\pi} \oint_c w(z)\,dz$$

2.3.2.5 Tabellen

`<TABLE>` `<TR>` `<TD>` `<TH>`

Tabellen werden von dem paarweise auftretenden *Tag* `<TABLE>` umschlossen. Die Zeilen beginnen mit `<TR>` (*Table Row*), die Zellen mit `<TD>` (*Table Data*). Für Zellen, die eine Überschrift enthalten, wird `<TH>` (*Table Header*) verwendet. Alle Marken treten paarweise auf. Der *Browser* fügt zwar eine fehlende Endemarke vor dem Auftreten einer neuen Marke automatisch ein, dennoch ist es empfehlenswert, alle Tabellenkonstruktionen immer abzuschließen.

`<CAPTION>` `ALIGN`

Mit der paarweise auftretenden Marke `<CAPTION>` wird für die Tabelle eine Überschrift gesetzt. Die Marke sollte innerhalb der Tabelle, vor dem Auftreten des ersten `<TR>`-*Tags*, eingefügt werden. Sie besitzt das Attribut `ALIGN`, wodurch ihre Ausrichtung in

Bezug auf die Tabelle (LEFT, RIGHT, CENTER (implizit)) festgelegt wird.

Das Aussehen einer Zelle kann durch Attribute verändert werden. Der Zelleninhalt kann durch ALIGN horizontal (LEFT, RIGHT, CENTER) bzw. durch VALIGN vertikal (TOP, BOTTOM, MIDDLE), positioniert werden. VALIGN

Die Breite bzw. Höhe einer Zelle kann mit den Attributen WIDTH und HEIGHT verändert werden. Die Angabe des Wertes erfolgt in Pixeln. Da eine leere Zelle oft nicht richtig dargestellt wird, sollte sie zumindest   enthalten. Des weiteren ist zu beachten, daß die manuelle Einstellung der Höhe bzw. Breite einer Zelle auch die Ausmaße aller anderen in der betreffenden Zeile und Spalte bestimmt. WIDTH HEIGHT

Mit dem Attribut NOWRAP kann verhindert werden, daß der Text in der Zelle an einer anderen Stelle als <BR> umgebrochen wird. NOWRAP

Die Breite und Höhe einer Zelle kann über mehrere Spalten bzw. Zeilen ausgedehnt werden. Dies geschieht mit Hilfe der Attribute COLSPAN und ROWSPAN. Mit COLSPAN wird bestimmt, wieviele Spalten für den Eintrag zusammengefaßt werden sollen. Wird ROWSPAN verwendet, so können in den Zeilen der Spalte, die von der ausgedehnten Zelle besetzt werden, keine anderen Einträge erfolgen (die Zellen werden weggelassen). COLSPAN ROWSPAN

Mit dem Attribut BGCOLOR kann die Hintergrundfarbe einer Zelle eingestellt werden. BGCOLOR

Alle gemeinsamen Attribute der Zellen einer Zeile werden im *Tag* <TR> angegeben. Es handelt sich dabei um die gleichen Attribute, wie bei <TD> bzw. <TH>, nämlich ALIGN, VALIGN und BGCOLOR. Die in den Zellen angegebenen Attribute haben gegenüber den globalen Vorrang.

Für <TABLE> gibt es ebenfalls das Attribut ALIGN. Diesmal handelt es sich jedoch um die Anordnung der Tabelle auf der Seite. Zulässige Werte sind wieder LEFT, RIGHT und CENTER. Im Gegensatz zu Bildern werden Tabellen nicht vom Text umflossen. ALIGN

Mit dem Attribut WIDTH kann die Gesamtbreite der Tabelle bestimmt werden. Der Wert wird entweder in Pixeln oder in Prozent angegeben. Im zweiten Fall wird festgelegt, wieviel Prozent der Gesamtbreite des Fensters die Tabelle einnehmen soll. WIDTH

Jede Zelle einer Tabelle hat man sich als Text, umgeben von einem leeren Rahmen, vorzustellen. Die Breite dieses Rahmens wird durch CELLPADDING bestimmt. Der Abstand (horizontal und vertikal) zwischen den gerahmten Zellen wird durch das Attribut CELLSPACING bestimmt. Die Werte beider Attribute werden in Pixeln angegeben. CELLPADDING CELLSPACING

Das Einrahmen der Tabelle wird durch das Attribut BORDER festgelegt. Implizit wird die Tabelle ohne Linien dargestellt (entspricht BORDER=0). Ist der Wert von BORDER größer als Null, so erhält die Tabelle einen Rahmen (der Dicke entsprechend dem Wert in Pixeln) und die einzelnen Zellen werden von Linien (der festen Breite 1 Pixel) eingerahmt. BORDER

Beispiel.

```
1  <TABLE CELLSPACING=15 CELLPADDING=0>
2  <TR><TD VALIGN=TOP>
3  <H2>Zweispaltiger Text</H2>
4  ist eine der m&ouml;glichen Anwendungen
5  f&uuml;r eine Tabelle. Der <EM>Client</EM>
```

```
6   versucht dabei nicht den rechten Rand
7   auszugleichen.<BR>
8
9      Von Nachteil ist, da&szlig;
10  kein automatisches Auff&uuml;llen der
11  Spalten erfolgt, dies mu&szlig; von
12  Hand geschehen.
13  </TD>
14
15  <TD VALIGN=TOP ALIGN=CENTER>
16  <H2>Beispiel f&uuml;r eine Tabelle</H2>
17  <TABLE BORDER=1 ALIGN=CENTER>
18
19  <TR ALIGN=CENTER>
20  <TH ROWSPAN=2>Monat</TH><TH COLSPAN=2>Erl&ouml;s</TH></TR>
21
22  <TR ALIGN=CENTER>
23
24  <!-- erste Zelle bleibt wegen ROWSPAN=2 leer -->
25
26  <TH>N&auml;gel</TH><TH>Schrauben</TH></TR>
27
28  <TR><TD>September</TD>
29      <TD ALIGN=RIGHT>2915</TD><TD ALIGN=RIGHT>1036</TD></TR>
30
31  <TR><TD>Oktober</TD>
32      <TD ALIGN=RIGHT>3058</TD><TD ALIGN=RIGHT>9764</TD></TR>
33
34  <TR BGCOLOR="#FFFF00"><TD>Summe</TD>
35      <TD ALIGN=RIGHT>5973</TD><TD ALIGN=RIGHT>10800</TD></TR>
36  </TABLE>
37  <DIV ALIGN=RIGHT>
38  <P>
39  <A HREF="mailto:Peter.Leibner@sta.siemens.de">Dr. Peter Leibner</A>
40  </DIV>
41  </TD></TR>
42  </TABLE>
```

Ergebnis → Abb.2.9

Mit Hilfe einer Tabelle wird der Text zweispaltig ausgegeben. Der obere Rand der Spalten wird durch die Angaben in den Zeilen 2 *und* 15 *ausgerichtet.*

*Die zweite Spalte enthält eine Tabelle, deren Eintrag „Monat“ sich über zwei Zeilen erstreckt. In der Zeile (*26*), welche die Einträge „Nägel“ und „Schrauben“ enthält, wird folglich die erste Zelle weggelassen.*

*Die Ergebniszeile (*34–35*) ist gelb unterlegt. Statt* `#FFFF00` *kann auch* `yellow` *angegeben werden.*

In der nachfolgenden Tabelle 2.15 sind einige vordefinierte Farben für die Verwendung

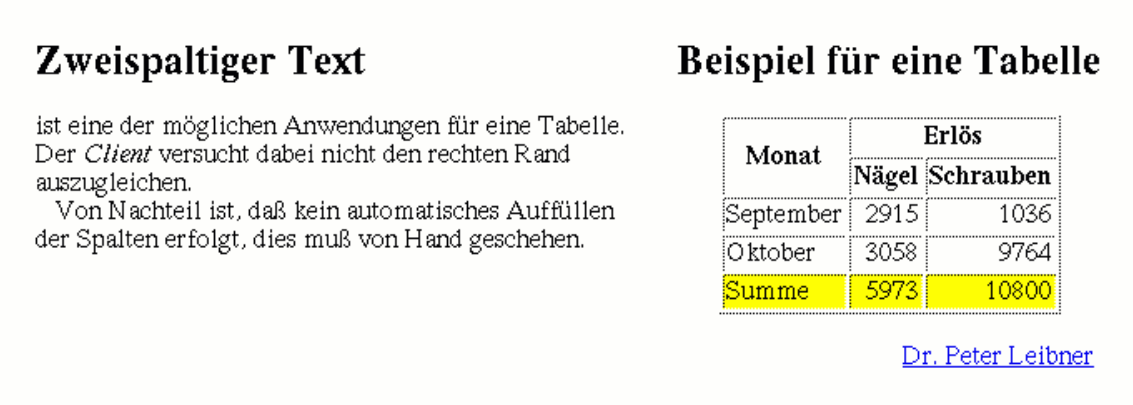


Abb. 2.9: Bildschirmausgabe, Bsp.2.3.2.5

in HTML-Seiten angegeben. Die Verwendung der vordefinierten Worte dürfte bei `red`*,* `yellow`*,* `white`*, usw., problemlos sein, bei* `maroon` *oder* `olive` *jedoch ein gewisses Risiko ob des gewünschten Effekts, darstellen.*

white	#FFFFFF	lime	#00FF00
silver	#C0C0C0	green	#008000
gray	#808080	yellow	#FFFF00
black	#000000	olive	#808000
red	#FF0000	blue	#0000FF
maroon	#800000	navy	#000080
fuchsia	#FF00FF	aqua	#00FFFF
purple	#800080	teal	#008080

Tab. 2.15: Vordefinierte Namen von Farben

2.3.2.6 Rahmen

Die Verwendung von Rahmen ist fragwürdig. Es gibt eine ganze Menge von Gründen, es nicht zu tun.

- Der aktuelle Inhalt einer gerahmten Seite kann nicht durch ein URL beschrieben und somit auch nicht als *Bookmark* abgelegt werden. Es wird immer nur die Ausgangsseite abgelegt, nicht jedoch eine durch den Benutzer veränderte.
- Komplette gerahmte Seiten lassen sich nicht ausdrucken, da nur der Inhalt des aktiven Rahmens ausgegeben wird.
- Aktiv ist immer der Rahmen, der zuletzt angeklickt wurde. Ändert sich dadurch der Inhalt eines anderen Rahmens, so kann dieser nur dann bearbeitet werden (z.B. verschoben werden), wenn dieser Rahmen zuvor angeklickt wird.
- Die Seiten müssen so gestaltet werden, daß auch *Browser*, die keine Rahmen unterstützen, sie darstellen können. Dies kann beträchtlichen Mehraufwand bedeuten.
- Eine gerahmte Seite wird aus mehreren Dateien zusammengesetzt. Zu ihrer Übertragung sind folglich mehrere HTTP-Verbindungen zum *Server* notwendig.

- Eine rahmenähnliche Seitengestaltung läßt sich auch mit Hilfe von Tabellen erreichen. Tabellen sind verbreiteter, einfacher und sicherer, daher auch empfehlenswerter.

<FRAMESET>

Bei der Verwendung von Rahmen entfällt die Marke `<BODY>`, sie wird durch `<FRAMESET>` und `</FRAMESET>` ersetzt. Innerhalb von `<FRAMESET>` dürfen nur Rahmen oder weitere `<FRAMESET>`-Umgebungen vorkommen, Text ist nicht erlaubt. Durch `<FRAMESET>` wird festgelegt, wie das Fenster aufgeteilt werden soll. Dazu ist eines der Attribute `ROWS` oder `COLS` zu verwenden. Mit `ROWS` wird die Seite in horizontale Rahmen zerlegt (Zeilen), mit `COLS` (Spalten) in vertikale.

ROWS
COLS

Die Werte von `ROWS` und `COLS` bestimmen die Größe der Rahmen. Sie können auf dreierlei Art und Weise angegeben werden (→ Tab.2.16).

Wert	Bedeutung
n	Fester Wert in Pixeln.
n%	Fester Wert in Prozent. Die Breite (Höhe) des Rahmens soll n% der aktuellen Breite (Höhe) des Fensters haben.
n*	Variabler Wert. Nach Festlegung der Ausmaße aller Rahmen mit festen Werten wird der restliche Platz im Verhältnis der Werte n den restlichen Rahmen zugeteilt. Fehlt die Angabe von n, so erfolgt die Aufteilung im gleichen Verhältnis.

Tab. 2.16: Eingabe der Werte für `ROWS` und `COLS`

Wird das Fenster nicht vollständig mit Hilfe von Prozentangaben (Summe: 100%) aufgeteilt, so sollte wenigstens ein Rahmen variabel gehalten werden. Erfolgen alle Angaben in Pixeln, so nimmt der *Browser* hinter jedem Wert einen Stern an und teilt seine Fensterfläche im Verhältnis der Werte auf.

Beispiel. *Die Angaben*

`<FRAMESET ROWS="25%,50%,25%">`,

`<FRAMESET ROWS="25%,*,25%">` *und*

`<FRAMESET ROWS="1*,2*,1*">`

sind alle gleichwertig. Sie teilen das aktuelle Fenster in drei horizontale Rahmen (drei Zeilen) auf, deren Höhe das Fenster im Verhältnis 1:2:1 teilt.

<FRAME>

Der Inhalt der Rahmen wird durch die Marke `<FRAME>` bestimmt. Es handelt sich dabei um eine nichtpaarweise vorkommende Marke, deren Attribute die Eigenschaften des Rahmens bestimmen. Zwingend vorgeschrieben ist nur `SRC`, das als Wert die URL der Datei enthält, die den Inhalt des Rahmens bilden soll. Es sollte sich dabei immer um eine HTML-Datei handeln.

SRC

SCROLLING
AUTO

Ein weiteres Attribut ist `SCROLLING`, wodurch bestimmt wird, ob der Rahmen einen Balken zum Verschieben des Inhalts erhalten soll oder nicht. Das Attribut kann die Werte `AUTO` (impliziter Wert, ein Balken wird erzeugt, wenn der Inhalt nicht in den

Rahmen paßt), YES (der Balken wird immer erzeugt) und NO (es wird kein Balken erzeugt), haben. Der Benutzer kann standardmäßig mit Hilfe der Maus die Größe des Rahmens verändern. Dies kann durch die Angabe des Attributs NORESIZE explizit verboten werden. Mit Hilfe der Attribute MARGINWIDTH und MARGINHEIGHT wird der Abstand zwischen Text und Rahmen bestimmt. Im ersten Fall handelt es sich um den waagrechten, im zweiten um den senkrechten Abstand.

YES
NO
NORESIZE
MARGINWIDTH
MARGINHEIGHT

Mit dem Attribut NAME kann dem Rahmen ein Name vergeben werden. Dieser wird zur Festlegung des Rahmens als Ausgabefenster verwendet. Der Verweis auf diesen Rahmen erfolgt über das Attribut TARGET in der Marke <A>.

NAME
TARGET

Um den ganzen Aufbau der gerahmten Seite zu beeinflussen, muß TARGET einer der speziellen Werte aus Tabelle 2.17 zugewiesen werden.

Wert	Bedeutung
_blank	es wird ein neues (namenloses) Fenster geöffnet
_top	die Rahmenstruktur wird vollständig zerstört, das Dokument wird im ganzen Fenster dargestellt (dies ist insbesondere dann sinnvoll, wenn der Verweis auf einen anderen *Server* geht)
_parent	die <FRAMESET>-Umgebung, zu welcher der aktuelle <FRAME> gehört, wird zerstört (ist dann sinnvoll, wenn z.B. wie im folgenden Beispiel 2.17 der Rahmen mit dem Firmenlogo erhalten bleiben soll)
_self	Darstellung des Dokuments im aktuellen Rahmen

Tab. 2.17: Spezielle Werte für TARGET

Beispiel. *Da für die Rahmendarstellung einer Seite mehrere aufeinander verweisende Dateien benötigt werden, ist ihr Aufbau in der Regel komplizierter als derjenige einer Seite ohne Rahmen. Die in diesem Beispiel verwendete Seite wird aus drei waagrechten Rahmen gebildet, wobei der mittlere nochmals in zwei senkrechte Rahmen zerlegt wird.*

Datei **beisp6.html***:*

```
1  <HTML>
2  <HEAD>
3  <TITLE>HTML in Beispielen -- Rahmen</TITLE>
4  </HEAD>
5  <FRAMESET ROWS="100,1*,40">
6  <FRAME SRC="nadpis.html" SCROLLING=NO>
7
8  <FRAMESET COLS="1*,100">
9  <FRAME SRC="unix.html" NAME="hauptrahmen">
10 <FRAME SRC="buecher.html" NAME="menu" SCROLLING=NO>
11 </FRAMESET>
12
13 <FRAME SRC="auswahl.html" SCROLLING=NO>
14 </FRAMESET>
15
```

```
16  <NOFRAMES>
17  <A HREF="buecher.html">B&uuml;cher</A><BR>
18  <A HREF="firma.html">Firmendaten</A>
19  </NOFRAMES>
20  </HTML>
```

Ergebnis → Abb.2.11

Die Tags `<NOFRAMES>` *(Zeile* 16*) und* `</NOFRAMES>` *(Zeile* 19*) werden von* Browsern*, die Rahmen unterstützen, ignoriert. Sie sind jedoch wichtig für* Clients*, die keine Rahmen darstellen können. Diese ignorieren wiederum alles, bis auf die beiden durch* `<NOFRAMES>` *geklammerten Zeilen, über welche sie Zugang zu den Dateien* **buecher.html** *und* **firma.html** *erhalten.*

Den Zusammenhang zwischen den einzelnen Rahmen zeigt Abbildung 2.10.

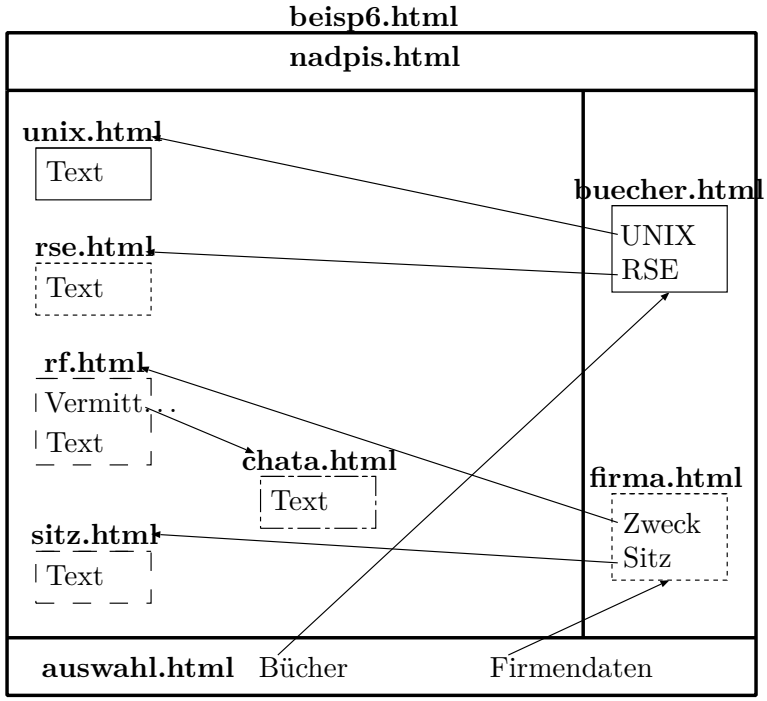


Abb. 2.10: Verbindungen im Fenster von Abb.2.11

Die Startseite enthält die Dateien **nadpis.html**, **unix.html**, **buecher.html** *und* **auswahl.html** *(Zeilen* 6, 9, 10 *und* 13 *— in Abb.2.10 mit durchgezogener Linie dargestellt).*

Datei **buecher.html***:*

```
1  <HTML>
2  <BODY>
3  <STRONG>B&uuml;cher</STRONG>
4  <P>
5  <A HREF="unix.html" TARGET="hauptrahmen">UNIX</A><BR>
6  <A HREF="rse.html" TARGET="hauptrahmen">RSE</A>
7  </BODY>
8  </HTML>
```

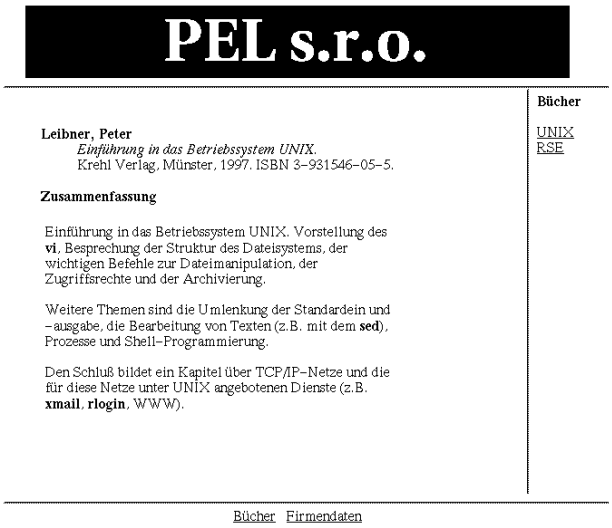


Abb. 2.11: Bildschirmausgabe, Bsp.2.17

Datei **unix.html***:*

```
1  <HTML>
2  <BODY>
3  <DL>
4  <DT><STRONG>Leibner, Peter</STRONG>
5  <DD><EM>Einf&uuml;hrung in das Betriebssystem UNIX.</EM><BR>
6  Krehl Verlag, M&uuml;nster, 1997. ISBN 3-931546-05-5.
7  </DL>
8  <STRONG>Zusammenfassung</STRONG>
9  <P>
10 <TABLE WIDTH=400>
11 <TR><TD>
12 Einf&uuml;hrung in das Betriebssystem UNIX.
13 Vorstellung des <B>vi</B>, Besprechung der
14 Struktur des Dateisystems, der wichtigen Befehle
15 zur Dateimanipulation, der Zugriffsrechte und
16 der Archivierung.
17 <P>
18 Weitere Themen sind die Umlenkung der
19 Standardein und -ausgabe, die Bearbeitung
20 von Texten (z.B. mit dem <B>sed</B>), Prozesse
21 und Shell-Programmierung.
22 <P>
23 Den Schlu&szlig; bildet ein Kapitel &uuml;ber
24 TCP/IP-Netze und die f&uuml;r diese Netze unter
25 UNIX angebotenen Dienste (z.B. <B>xmail</B>,
26 <B>rlogin</B>, WWW).
27 </TD></TR></TABLE>
28 </BODY>
29 </HTML>
```

Datei **rse.html***:*

```
1   <HTML>
2   <BODY>
3   <DL>
4   <DT><STRONG>Leibner, Peter</STRONG>
5   <DD><EM>Rechnergest&uuml;tzter Schaltungsentwurf.</EM><BR>
6   Krehl Verlag, M&uuml;nster, 1997. ISBN 3-931546-02-0.
7   </DL>
8   <STRONG>Zusammenfassung</STRONG>
9   <P>
10  <TABLE WIDTH=400>
11  <TR><TD>
12  Herleitung der mathematischen Grundlagen zur
13  Beschreibung digitaler und analoger Schaltungen.
14  <P>
15  Entwurf kombinatorischer und sequentieller Schaltungen.
16  Logiksimulation und Testmustergenerierung.
17  <P>
18  Analyse linearer, nichtlinearer und dynamischer
19  analoger Schaltungen. Laplace- und Z-Transformation.
20  Numerische Integrationsverfahren zur
21  Berechnung dynamischer Netzwerke.
22  </TD></TR></TABLE>
23  </BODY>
24  </HTML>
```

Datei **auswahl.html***:*

```
1   <HTML>
2   <BODY>
3   <DIV ALIGN=CENTER>
4   <A HREF="buecher.html" TARGET="menu">B&uuml;cher</A>  
5   <A HREF="firma.html" TARGET="menu">Firmendaten</A>
6   </DIV>
7   </BODY>
8   </HTML>
```

Nach Anklicken von `Firmendaten` *im unteren Rahmen (Datei* **auswahl.html***) wird in den rechten mittleren Rahmen die Datei* **firma.html** *geladen.*

Datei **firma.html***:*

```
1   <HTML>
2   <BODY>
3   <STRONG>Firma</STRONG>
4   <P>
5   <A HREF="rf.html" TARGET="hauptrahmen">Zweck</A><BR>
```

```
6   <A HREF="sitz.html" TARGET="hauptrahmen">Sitz</A>
7   </BODY>
8   </HTML>
```

Von dieser können in den großen mittleren Rahmen die Dateien **rf.html** *und* **sitz.html** *geladen werden.*

Datei **sitz.html***:*

```
1   <HTML>
2   <BODY>
3   <H1>Sitz der PEL s.r.o.</H1>
4
5   <ADDRESS>
6   PEL s.r.o. (ICO: 25 03 50 96)<BR>
7   c/o Jiri Vinduska<BR>
8   Pod Bornym 0249<BR>
9   CZ-47163 Stare Splavy<BR>
10  Ceska republika
11  </ADDRESS>
12  </BODY>
13  </HTML>
```

Datei **rf.html***:*

```
1   <HTML>
2   <BODY>
3   <H1>PEL s.r.o.</H1>
4   <UL>
5   <LI>Gesellschaft mit beschr&auml;nkter Haftung
6   <LI>Zweck
7   <OL>
8   <LI><A HREF="chata.html" TARGET="_self">
9     Vermittlungst&auml;tigkeit</A>
10  <LI>An- und Verkauf von Waren mit Ausnahme von Arzneimitteln,
11  Drogerieartikeln und konzessierten Waren
12  </OL>
13  </UL>
14  </BODY>
15  </HTML>
```

Aus dem mittleren Rahmen kann weiter aus der Datei **rf.html** *die Datei* **chata.html** *in denselben Rahmen geladen werden* (`TARGET="_self"`).

Datei **chata.html***:*

```
1   <HTML>
2   <BODY>
```

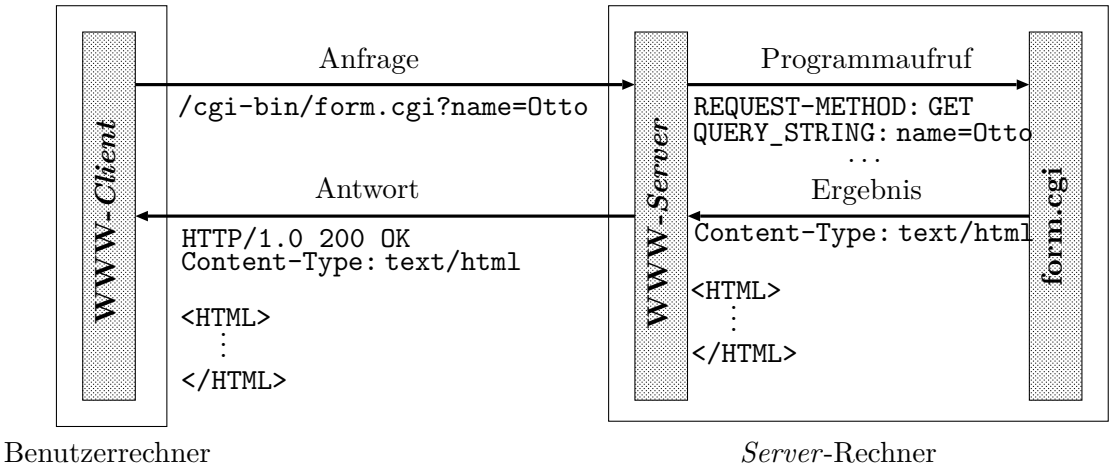


Abb. 2.12: Arbeitsweise von CGI

```
3   <H1>Vermittlung</H1>
4   des Ferienbungalows c.ev. 33, parcela c. 257,
5   katastralni uzemi obce Predni Vyton,
6   38279 Frymburg, okres Cesky Krumlov, Ceska republika.
7   </BODY>
8   </HTML>
```

FRAMEBORDER

Will man die Grenzen der Rahmen nicht dargestellt haben, so ist der Marke `<FRAME>` *das Attribut* `FRAMEBORDER="0"` *hinzuzufügen.*

2.3.2.7 CGI

Das *Common Gateway Interface* ermöglicht es, an einem *Server* ein durch die URL spezifiziertes Programm zu starten und das Ergebnis an den *Client* zurückzuschicken.

CGI ist eine Schnittstelle zwischen dem WWW-*Server* und einem CGI-Programm[43]. Bevor der *Server* das Programm startet, speichert er Informationen aus der Anfrage sowie über die Ablaufumgebung in Umgebungsvariablen (→ Tab.2.18). Danach startet er das Programm. Dieses gibt sein Ergebnis auf die Standardausgabe aus, wo es der *Server* entgegen nimmt, mit Statusinformationen versieht und als Antwort an den *Client* schickt (→ Abb.2.12).

Das Programm muß sich folglich nicht um die Kommunikation mit dem *Client* kümmern, dies besorgt der *Server*.

Wie bereits erwähnt (→ S.68), hat eine Anfrage eine ähnliche Form wie ein Brief der elektronischen Post.

methode weg HTTP_version
kopf

[43] auch CGI-Skript genannt

rumpf

Sie enthält Angaben über die *methode* (`GET` oder `POST`) und den *weg* zu den gewünschten Informationen (in der Regel einem Programm namens *programm*.**cgi**). Um welche *methode* es sich handelt, erfährt das CGI-Programm aus den Umgebungsvariablen. Die wichtigsten sind in Tabelle 2.18 zusammengesfaßt.

Umgebungsvariable	Bedeutung
`REQUEST_METHOD`	gibt die verwendete *methode* an (`GET` oder `POST`)
`QUERY_STRING`	enthält die Anfrage des *Clients*, also den Teil, der dem Fragezeichen folgt (→ Bsp.2.7)
`CONTENT_LENGTH`	ist die Länge der Anfrage im *rumpf* (in Byte), falls die *methode* `POST` (`REQUEST_METHOD=POST`) verwendet wurde
`CONTENT_TYPE`	gibt bei der *methode* `POST` an, welcher Art die übergebenen Daten sind
`PATH_INFO`	Weg zur Datei aus der URL (→ Bsp.2.18)
`PATH_TRANSLATED`	Weg zur Datei im Dateisystem (→Bsp.2.18)
`SCRIPT_NAME`	URL des CGI-Programms (→ Bsp.2.18)

Tab. 2.18: CGI-Umgebungsvariablen

Beispiel. *Bei der URL* `http://www.firma.de/cgi-bin/imagemap/img/bild.map` *ist das CGI-Programm* **imagemap** *und* `PATH_INFO=/img/bild.map` *die zu bearbeitende Datei.*

Falls das Wurzelverzeichnis auf dem Server `/net/www/html` *sein sollte, so folgt für* `PATH_TRANSLATED=/net/www/html/img/bild.map`.

Die URL des CGI-Programms ist `SCRIPT_NAME=/cgi-bin/imagemap`.

Die am häufigsten verwendete Methode ist `GET`, bei welcher der *rumpf* leer bleibt. Dabei erhält der *Server* die Anfrage über die URL. Die Übergabe der Anfrage an das CGI-Programm erfolgt entweder als Argumente beim Programmaufruf oder über die Umgebungsvariable `QUERY_STRING` (→ Abb.2.12[44]). Die Menge der Informationen, die auf diese Art und Weise übergeben werden kann, ist auf ca. 200 Byte beschränkt. Da sie mit der URL übergeben werden, tauchen sie auch im Protokoll über die *Server*-Tätigkeit auf, was bei vertraulichen Informationen sicher nicht erwünscht sein kann. `GET`

Bei der Methode `POST` erhält der *Server* die Anfrage im *rumpf*. Die Leerzeile hinter dem *kopf* (der aus mehreren Kopfzeilen bestehen kann), muß immer vorhanden sein, auch wenn der *rumpf* leer sein sollte. Laut CGI-Definition darf das Programm aus der Standardeingabe nicht mehr als `CONTENT_LENGTH`-Byte einlesen. Ob der *rumpf* z.B. ein Bild, Text oder ein HTML-Dokument enthält, erfährt das Programm aus dem Wert von `CONTENT_TYPE`. `POST`

Sowohl bei `GET` als auch bei `POST` sind die Daten normalerweise URL-kodiert (→ S.71). Um die ursprüngliche Form zu erhalten, muß das CGI-Programm die Sonderzeichen aus den Daten wieder entfernen.

[44] es wird das CGI-Programm **form.cgi** aufgerufen, das die Daten aus der Umgebungsvariablen `QUERY_STRING` erhält

Die HTTP-Version 1.1 kennt noch weitere Methoden, wie `HEAD` (bereits in 1.0 vorhanden), `PUT`, `DELETE`, `OPTIONS` und `TRACE`.

Die HTTP-Antwort sieht der Anfrage sehr ähnlich.

HTTP_version status beschreibung
kopf

rumpf

Der ersten Zeile kann der *Client* entnehmen, ob die Anfrage erfolgreich war oder nicht. Dies erfährt er aus dem *status* und der *beschreibung*. Der *status* ist ein numerischer Kode, gedacht für den *Client*, die *beschreibung* ist eine kurze Beschreibung des Kodes für den Benutzer.

Der *status* wird durch drei Ziffern angegeben, die unterschiedliche Gruppen definieren. Einen Erfolg signalisieren Angaben der Form `2`*xx* (`OK`) (→ Abb.2.12).

Die Gruppe `3`*xx* gibt an, daß das gewünschte Objekt an eine andere Stelle verlegt wurde. Der *kopf* muß dann die Zeile `Location`[45] enthalten, in der die neue URL stehen muß.

Beispiel.

```
HTTP/1.0 301 Moved Permanently
Location: http://www.woanders.de/datei.html
Content-Type: text/html

Die Datei hat eine neue
<A HREF="http://www.woanders.de/datei.html">URL</A>
```

Es wird empfohlen, dem Benutzer einen Verweis auf die neue URL zu schicken. Moderne Clients *sollten jedoch ohne diesem Hinweis auskommen und die Anfrage automatisch an die neue URL wiederholen.*

Die Gruppe `4`*xx* signalisiert einen Fehler auf Seiten des *Clients*. Die häufigste Fehlermeldung ist `404` (`Not Found`), was ein Hinweis darauf ist, daß die URL nicht korrekt angegeben wurde.

In die Gruppe `5`*xx* fallen Fehler des *Servers*. Diese treten in der Regel nur selten auf.

Beispiel. *Der* Client *stellt zum* Server *eine Verbindung her und schickt ihm folgende Anfrage:*

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.01 [en] (Win95; I)
Host: www.firma.de
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

[45] Die „Header“ `Content-Type` und `Location` (sowie gegebenenfalls `Status`) muß das CGI-Programm einfügen. Die Angabe `Content-Type` ist verpflichtend, die beiden anderen optional.

Leere Zeile

In der Anfrage hat der Client *die Titelseite (der* weg *ist* **/***) angefordert. Der* Server *antwortet wie folgt:*

```
HTTP/1.0 200 OK
Date: Wed, 14 Jan 1998 15:16:55 GMT
Server: Apache/1.2.6
Content-Type: text/html
Content-Length: 1486
Last-Modified: Mon, 02 Jun 1996 11:37:47 GMT

<HTML>
  .
  .
</HTML>
```

In der Statusangabe wird dem Client *mitgeteilt, daß die Anfrage erfolgreich bearbeitet werden konnte.*

Eine der besten Sprachen zur Auswertung von Anfragen ist Perl[46]. Perl

Im Vorgriff auf den nächsten Unterabschnitt, der sich Formularen widmet, wird im folgenden Beispiel die Auswertung einer Anfrage mit Hilfe eines in Perl geschriebenen CGI-Programms gezeigt.

Beispiel. *Mit der Methode* `GET` *sollen aus einer Datei* **telefon.txt** *nach Eingabe eines Namens die zugehörigen Zeilen ausgegeben werden. Jede Zeile enthält den Namen, ein Tabulatorzeichen und die Telefonnummer, z.B.*

```
Fr. Heiss  +49 (0)89 722 2 73 72
Fr. Sczesny  +49 (0)89 722 2 58 20
Fr. Kraus  +49 (0)89 722 2 68 44
Dr. Ziegler  +49 (0)172 8 55 85 49
Dr. Leibner  +49 (0)89 722 6 34 36
Dr. Leibner  +49 (0)89 722 3 61 35 31
Dr. Leibner  +49 (0)172 8 61 20 27
```

*Die zugehörige HTML-Seite (***telefon.html***) enthält ein einfaches Formular:*

```
1  <HTML>
2  <HEAD>
3  <TITLE>Telefonverzeichnis</TITLE>
4  </HEAD>
5  <BODY BGCOLOR=white>
6  <P>
```

[46] *Practical Extraction and Report Language* – Auswahl von Informationen und ihre Ausgabe – alternativer Name von ihrem Entwickler Larry Wall: *Pathologically Eclectic Rubbish Lister*

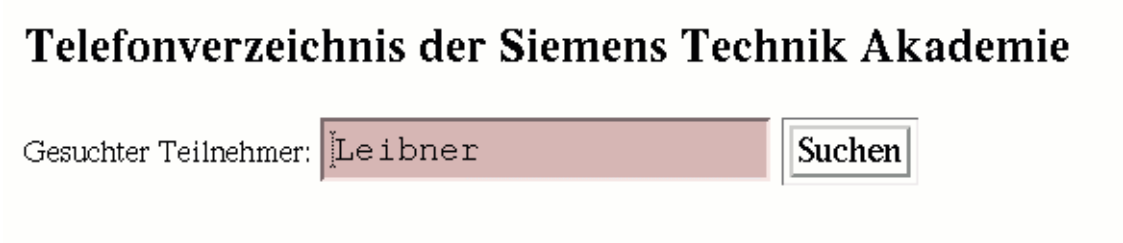


Abb. 2.13: Bildschirmausgabe, Bsp.2.18

```
7   <H1>Telefonverzeichnis der Siemens Technik Akademie</H1>
8   <P>
9   <FORM ACTION="suchen.cgi" METHOD=GET>
10  Gesuchter Teilnehmer: <INPUT NAME="name">
11  <INPUT TYPE=SUBMIT VALUE="Suchen">
12  </FORM>
13  </BODY>
14  </HTML>
```

Ergebnis → Abb.2.13

Die wichtigsten Angaben stehen in Zeile 9 *— die Methode* `GET` *und das CGI-Programm* **suchen.cgi**, *dem das Formular zur Auswertung vorgelegt werden soll. Es liegt im gleichen Verzeichnis wie die Datei* **telefon.html** *und ist in Perl geschrieben.*

```
1   #!/usr/local/bin/perl
2   # CGI-Programm suchen.cgi
3
4   $dataname = "telefon.txt";    # name der Datei
5
6   # gesuchte Zeichenkette ermitteln, ueberfluessige Zeichen loeschen
7   $suchen = $ENV{"QUERY_STRING"};
8   $suchen =~ s/name=//;
9   $suchen =~ s/[^a-zA-Z0-9 \.]/ /g;
10
11  # Kopf und Anfang des Rumpfes
12  print "Content-Type: text/html\n\n";
13  print "<HTML>\n<HEAD>\n";
14  print "<TITLE>Ergebnis der Suche nach $suchen</TITLE>\n";
15  print "</HEAD>\n<BODY>\n\n";
16  print "<H1>Ergebnis der Suche nach $suchen</H1>\n";
17
18  # Suche und Ergebnisausgabe
19  open ( DATA, $dataname );
20  while ( $radek = <DATA> ) {
21     chop ( $radek );
22     if ( $radek =~ /$suchen/i ) {
23        $radek =~ s/\t/ ... Verbindung /;
24        print "$radek<BR>\n";
25        $gefunden++;
```

```
26      }
27   }
28   close ( DATA );
29
30   # Seitenende
31   if ( $gefunden == 0 ) {
32      print "Zeichenkette nicht gefunden.\n";
33   } else {
34      print "<P>H&auml;ufigkeit des Auftretens: $gefunden\n";
35   }
36   print "</BODY>\n</HTML>\n";
```

In Zeile 7 *wird aus der Umgebungsvariablen* `QUERY_STRING` *die Zeichenkette, die in der URL dem Fragezeichen folgt, in der Variablen* `$suchen` *gespeichert.*

Aus der URL `http://www.sta.siemens.de/~leibner/suchen.cgi?name=Leibner` *folgt bei einer Eingabe entsprechend Abbildung 2.13 für die Zeichenkette* `name=Leibner`*. In Zeile* 8 *wird aus ihr* `name=` *entfernt und in Zeile* 9 *werden alle Zeichen, die nicht Buchstabe, Ziffer, Leerzeichen oder Punkt sind, aus Sicherheitsgründen durch ein Leerzeichen ersetzt.*

In den Zeilen 12–16 *wird der Kopf und der Anfang des Ergebnisses erzeugt.*

In Zeile 19 *wird die Datei* **telefon.txt** *geöffnet, Zeile für Zeile gelesen und mit der gesuchten Zeichenkette verglichen (Zeile* 22*). Wird sie gefunden, so wird das Tabulatorzeichen durch* `...` `Verbindung` *ersetzt und die Zeile auf die Standardausgabe ausgegeben. Danach wird die Häufigkeit, mit der die Zeichenkette aufgetreten ist, in der Variablen* `$gefunden` *um eins erhöht (Zeile* 25*).*

In Zeile 28 *wird die Datei* **telefon.txt** *wieder geschlossen und die HTML-Seite beendet (Zeilen* 31–36*).*

Das Programm geht davon aus, daß sowohl in der gesuchten Zeichenkette als auch in den Eingabezeilen aus **telefon.txt** *nur Standard-ASCII-Kode für die Zeichen verwendet wird. Sollten Sonderzeichen vorkommen (→ S.71), so könnten sie durch folgende Zeilen entfernt werden:*

```
$radek =~ s/+/ /g;
$radek =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex(\1))/eq;
```

Die Reihenfolge der Zeilen ist wichtig, da sonst eventuell Pluszeichen, die durch das Dekodieren von `%`XY *entstehen, durch Leerzeichen ersetzt würden.*

Soll anstelle von `GET` *die Methode* `POST` *verwendet werden, so muß lediglich Zeile* 7 *durch*

```
read ( STDIN, $suchen, $ENV{"CONTENT_LENGTH"} );
```

ersetzt werden. Dann werden die Daten nicht mehr aus der Umgebungsvariablen sondern von der Standardeingabe eingelesen.

Die Antwort des Servers *zeigt Abb.2.14.*

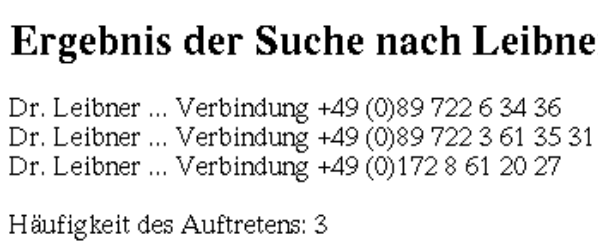


Abb. 2.14: Bildschirmausgabe, Antwort auf die Eingabe in Abb.2.13

CGI-Programme werden für eine Reihe von Anwendungen verwendet, wie z.B. zum Zählen der Besucher einer Seite, der Änderung der Zeichenkodierung (z.B. dem Umwandeln von ř → r, ů → u, usw.) oder dem Zugriff auf Datenbasen. Für alle diese Applikationen gibt es bereits fertige Produkte[47].

Abbildung 2.12 zeigt, daß die CGI-Programme ihre Standardausgabe direkt an den WWW-*Server* übergeben. Da die *Server* beim Auftreten von Fehlern keine Angaben über die Ursachen machen, ist es meistens nicht ohne weiteres möglich, eine fehlerhafte Ausgabe des CGI-Programms nachzuvollziehen.

Es gibt jedoch die Möglichkeit, für das CGI-Programm eine Ablaufumgebung zu emulieren und es dann direkt zu starten. Die Standardausgabe erfolgt dann auf den Bildschirm, ein Fehler im Programm kann dadurch sichtbar gemacht und behoben werden.

Beispiel. *Um ein ablauffähiges Skript zum Testen des CGI-Programms aus Beispiel 2.18 zu bekommen, ist in der Datei* **telefon.html** *das Programm* **suchen.cgi** *durch* **emcgiunx.cgi** *(das im gleichen Verzeichnis wie* **suchen.cgi** *stehen muß) zu ersetzen.*

Der Inhalt von **emcgiunx.cgi** *ist folgender:*

```
#!/usr/local/bin/perl

$tmpfile = "/tmp/cgi-emu.$$";

# print header and first line of emulator
print "Content-Type: text/plain\n\n";
print "#!/bin/sh\n\n";

# modify environment variables
$ENV{"SCRIPT_NAME"} =~ s/cgiemu$/\$CGIPROGRAM/;
$ENV{"SCRIPT_FILENAME"} =~ s/cgiemu$/\$CGIPROGRAM/;

# set up environment variables
foreach $var ( sort ( keys %ENV ) ) {
  if ( $var =~ /^SCRIPT/i ) {
    print "$var=\"$ENV{$var}\"\n"; }
  else {
```

[47] `http://hoohoo.ncsa.uiuc.edu/cgi/` (nicht nur Perl) und `http://cgi-lib.stanford.edu/cgi-lib/` für CGI-Programme in Perl


```
    print "$var='", $ENV{$var}, "'\n"; }
  print "export $var\n";
}
print "\n";

if ( $ENV{"REQUEST_METHOD"} =~ /GET/i ) {
  print "./\$CGIPROGRAM\n"
}
elsif ( $ENV{"REQUEST_METHOD"} =~ /POST/i ) {
  print "./\$CGIPROGRAM < $tmpfile\n";
  open ( TMP, ">$tmpfile" );
  read ( STDIN, $data, $ENV{"CONTENT_LENGTH"} );
  print TMP $data;
  close ( TMP );
}
```

Schickt man das geänderte Formular ab, so erhält man als Antwort z.B. folgendes:

```
#!/bin/sh

DOCUMENT_ROOT='/usr/local/apache/htdocs'
export DOCUMENT_ROOT
GATEWAY_INTERFACE='CGI/1.1'
export GATEWAY_INTERFACE
HTTP_ACCEPT='image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*'
export HTTP_ACCEPT
HTTP_CONNECTION='Keep-Alive'
export HTTP_CONNECTION
HTTP_HOST='www.sta.siemens.de'
export HTTP_HOST
HTTP_REFERER='http://www.sta.siemens.de/~leibner/telefon.html'
export HTTP_REFERER
HTTP_USER_AGENT='Mozilla/3.04 (X11; I; Linux 2.0.30 i586)'
export HTTP_USER_AGENT
PATH='/sbin:/usr/sbin:/usr/bin'
export PATH
QUERY_STRING='name=Leibner'
export QUERY_STRING
REMOTE_ADDR='172.21.184.5'
export REMOTE_ADDR
REMOTE_HOST='antje.sta.siemens.de'
export REMOTE_HOST
REMOTE_PORT='17717'
export REMOTE_PORT
REQUEST_METHOD='GET'
export REQUEST_METHOD
REQUEST_URI='/~leibner/emcgiunx.cgi?name=Leibner'
```

```
export REQUEST_URI
SCRIPT_FILENAME="/usr/home/leibner/public_html/emcgiunx.cgi"
export SCRIPT_FILENAME
SCRIPT_NAME="/~leibner/emcgiunx.cgi"
export SCRIPT_NAME
SERVER_ADMIN='webmaster@sta.siemens.de'
export SERVER_ADMIN
SERVER_NAME='zeta.zfe.siemens.de'
export SERVER_NAME
SERVER_PORT='80'
export SERVER_PORT
SERVER_PROTOCOL='HTTP/1.0'
export SERVER_PROTOCOL
SERVER_SOFTWARE='Apache/1.2.6'
export SERVER_SOFTWARE

./$CGIPROGRAM
```

Dies kann z.B. in die Datei **testget** *geschrieben, die Datei in das Verzeichnis, in dem die CGI-Programme liegen, kopiert und durch*

`~/public_html#` **chmod a+x testget**

ablauffähig gemacht werden. Jetzt muß noch der Umgebungsvariablen `CGIPROGRAM` *durch*

`~/public_html#` **export CGIPROGRAM=suchen.cgi**

der Name des ursprünglichen CGI-Programms zugewiesen und das Skript **testget** *gestartet werden. Die Bildschirmausgabe sieht dann wie folgt aus:*

```
Content-Type: text/html

<HTML>
<HEAD>
<TITLE>Ergebnis der Suche nach Leibner</TITLE>
</HEAD>
<BODY>

<H1>Ergebnis der Suche nach Leibner</H1>
Dr. Leibner ... Verbindung  +49 (0)89 722 6 34 36<BR>
Dr. Leibner ... Verbindung  +49 (0)89 722 3 61 35 31<BR>
Dr. Leibner ... Verbindung  +49 (0)172 8 61 20 27<BR>
<P>H&auml;ufigkeit des Auftretens: 3
</BODY>
</HTML>
```

Dies ist korrekt, wie man ja auch aus Abbildung 2.14 entnehmen kann.

Wird als Methode `POST` *verwendet, so erzeugt* **emcgiunx.cgi** *aus der Standardeingabe die Datei* **/tmp/emc-emu.**PID*. Dem CGI-Programm wird diese Datei dann im Skript,*

das wieder aus der Server-*Antwort erzeugt wird und jetzt z.B.* **testpost** *heißen kann, als Standardeingabe übergeben. Die Variable* `CGIPROGRAM` *muß wieder mit dem Namen der Datei* **suchen.cgi** *belegt werden, in welcher die Zeile* 7 *durch*

```
read ( STDIN, $suchen, $ENV{"CONTENT_LENGTH"} );
```

zu ersetzen ist.

In **telefon.html** *muß die Methode* `POST` *eingetragen sein.*

2.3.2.8 Formulare

Es gibt insgesamt vier grundlegende Methoden, wie einem CGI-Programm Informationen übergeben werden können.

Fester Verweis. Die URL endet mit `?`*daten*. Der Benutzer hat keine Möglichkeit, die *daten* selbst zu modifizieren.

Bild mit Klinke. Diese Bilder (nächster Abschnitt) werden durch die Marke `<IMG>` zusammen mit dem Attribut `ISMAP` realisiert. Sie dienen lediglich dazu, dem *Server* die Bildkoordinaten mitzuteilen, auf die der Benutzer mit seiner Maus geklickt hat[48]

Die Marke `<ISINDEX>`. Eine einfache Alternative zum Formular. Sie macht das CGI-Programm jedoch komplizierter. Dieses muß nämlich unterschiedlich reagieren, wenn es mit einer Anfrage (`?`*daten*) aufgerufen wird oder ohne. Wird es ohne Anfrage aufgerufen[49], so muß es mit einer HTML-Seite antworten, welche die Marke `<ISINDEX>` entweder zwischen `<HEAD>` und `</HEAD>` oder im Rumpf enthält[50]. Der *Client* stellt diese Seite dann mit einem einfachen Eingabefenster dar. Füllt der Benutzer das Fenster aus und schickt es an den *Server*, dann bekommt dieser die gleiche Anfrage wie vorher, diesmal jedoch mit dem Fragezeichen und den Daten am Ende. Jetzt muß das CGI-Programm diese Anfrage analysieren und entsprechend reagieren. <ISINDEX>

Die Tatsache, daß sowohl die HTML-Seite mit `<ISINDEX>` als auch die Antwort auf das ausgefüllte Formular vom selben CGI-Programm erstellt werden müssen, erschwert nicht nur die Erstellung des Programms, sondern verhindert auch, daß die Seite im Ausgleichsspeicher abgelegt werden kann. Das nur einzeilige Eingabefenster schränkt außerdem die Möglichkeiten des Benutzers, mit dem *Server* zu kommunizieren, stark ein.

Es wird daher empfohlen, diesen *Tag* nicht zu verwenden.

Formular. Ein Formular kann bedarfsgerecht zusammengestellt werden. Zur Übertragung der Daten stehen die Methoden `GET` und `POST` zu Verfügung. Die HTML-Seite mit einem oder mehreren Formularen ist zudem rein statisch und kann daher

[48] Der englische Begriff *clickable image* ist wenig aussagekräftig, schließlich kann man ja jedes Bild anklicken. Ich habe den Begriff „Bild mit Klinke“ aus [13] übernommen, da er meiner Meinung nach am besten wiedergibt, was nach einem Anklicken geschieht — es wird in Abhängigkeit vom Ort auf dem Bild eine neue Seite geladen (quasi eine Tür geöffnet).

[49] z.B. über den Verweis `http://www.firma.de/cgi-bin/anfrage.cgi`

[50] Steht sie im Rumpf, so kann sie mit erklärendem Text umgeben werden.

in einem Ausgleichsspeicher abgelegt werden, so daß sie nicht bei jeder erneuten Anfrage vom *Server* neu übertragen werden muß.

<FORM>

Das Formular ist Teil einer normalen HTML-Seite. Es wird durch die paarweise auftretende Marke `<FORM>` eingeschlossen. Es ist zwar möglich, auf jede Seite mehr als nur ein Formular zu plazieren, dies macht aber in der Regel keinen Sinn und es erschwert zudem die Orientierung auf der Seite.

ACTION

Der *Tag* `<FORM>` hat mehrere Attribute. Das wichtigste ist sicher `ACTION`, dessen Wert die URL angibt, an die das Formular geschickt werden soll. Dies kann z.B. die URL des zu verarbeitenden CGI-Programms (→ Bsp.2.18) sein oder die *e-mail*-Adresse, an die das Formular geschickt werden soll (→ Bsp.2.3.2.8).

METHOD

Ebenso wichtig wie `ACTION` ist das Attribut `METHOD`. Es kann die Werte `GET` und `POST` annehmen.

ENCTYPE

Ein weiteres, ebenso wichtiges Attribut, ist `ENCTYPE` (*Encoding-Type*). Der Standardwert von `ENCTYPE` ist `application/x-www-form-urlencoded`.

Beispiel. *Schickt ein Benutzer ein ausgefülltes Formular, z.B. mit elektronischer Post, an einen Empfänger, so erhält dieser bei beiden Methoden (*`POST` *und* `GET`*) z.B. folgendes:*

```
NAME=Leibner&VNAME=Peter&EMAIL=Peter.Leibner@pd.siemens.de&TEXT=Bitte+um%0D%0AAntwort.
```

Die e-mail *ist URL-kodiert und muß nachbearbeitet werden, um wieder die ursprüngliche Form zu erhalten (→ CGI-Programm).*

Die URL-Kodierung kann vermieden werden, wenn das Attribut `ENCTYPE` mit dem Wert `text/plain` der Marke `<FORM>` hinzugefügt wird.

Beispiel. *Für*

```
<FORM  METHOD="POST" ACTION="mailto:Peter.Leibner@pd.siemens.de" ENCTYPE="text/plain">
```

folgt für die gleiche Eingabe wie im vorhergehenden Beispiel jetzt

```
NAME=Leibner
VNAME=Peter
EMAIL=Peter.Leibner@pd.siemens.de
TEXT=Bitte um
Antwort.
```

NAME

Der Text zwischen `<FORM>` und `</FORM>` wird aus normalem HTML-Text und Formularmarken gebildet. Alle Formularmarken müssen das Attribut `NAME` enthalten, anhand dessen das CGI-Programm erkennen kann, in welches Feld der Benutzer die Anfrage eingegeben hat, bzw. welcher Teil des Formulars aktiv ist.

<INPUT>
TYPE

Die wichtigste Marke innerhalb einer `<FORM>`-Umgebung ist die nichtpaarweise vorkommende Marke `<INPUT>`. Mit Hilfe dieses *Tags* können unterschiedliche Eingabefelder definiert werden. Die Art des Feldes wird durch das Attribut `TYPE` bestimmt.

TEXT

Am häufigsten werden Fenster zur Eingabe von Zeichenketten verwendet. Implizit ist daher `TYPE=TEXT` eingestellt.

Zwei weitere Attribute bestimmen die Größe des Fensters und die maximale Länge der Eingabe.

Mit MAXLENGTH wird die maximale Länge der Zeichenkette festgelegt. Der *Client* erlaubt dann keine längeren Eingaben. MAXLENGTH

Mit SIZE wird bestimmt, wieviele Zeichen lang das Eingabefenster sein soll. Gibt der Benutzer mehr Zeichen ein, so verschiebt der *Client* den Anfang der Eingabe nach links, wodurch dieser unsichtbar wird aber nicht verloren geht. SIZE

Mit dem Attribut VALUE kann das Eingabefenster mit einer Zeichenkette voreingestellt werden. VALUE

Der TYPE=PASSWORD entspricht dem vorhergehenden impliziten TYPE=TEXT, der eingegebene Text wird jedoch vom *Client* nicht dargestellt. Die Übertragung erfolgt aber unchiffriert, wodurch die Eingabe zu einem gewissem Sicherheitsrisiko wird. PASSWORD

Häufig treten in einem Formular ein oder mehrere Schalter auf. Der TYPE wird dabei auf CHECKBOX gesetzt. Das Attribut NAME dient zur Identifikation des Schalters, mit Hilfe von VALUE kann man vorschreiben, welche Daten an den *Server* geschickt werden sollen, wenn der Schalter eingeschaltet wird. Ist er ausgeschaltet, so werden keine Daten an den *Server* übertragen. Mit dem Attribut CHECKED erhält der Schalter den impliziten Status „eingeschaltet“. CHECKBOX CHECKED

Es ist üblich, Schalter, die einer gemeinsamen Aufgabe dienen, unter dem gleichen NAME zusammenzufassen. Sie werden dann durch die unterschiedliche Werte von VALUE identifiziert.

Eine ähnliche Funktion wie der Schalter hat der Umschalter. Er setzt sich aus mehreren *Tags* der Form <INPUT TYPE=RADIO> zusammen. Sie haben alle einen gemeinsamen NAME, unterscheiden sich jedoch in den Werten von VALUE. Da es sich um Umschalter handelt, kann immer nur einer aktiv sein. Sein Einschalten erfolgt mit dem Attribut CHECKED. RADIO

Ein weiteres Element eines Formulars ist der Knopf, mit dessen Hilfe das Formular abgeschickt werden kann. Der TYPE bekommt dazu den Wert SUBMIT zugewiesen. Der Wert des Attributs VALUE bestimmt die Aufschrift, die der Knopf erhält. SUBMIT

Der Knopf muß nicht zwangsweise das Attribut NAME enthalten. Dieses muß nur dann vorhanden sein, wenn das Formular mehrere Knöpfe zum Verschicken enthält und das CGI-Programm je nach gedrücktem Knopf entscheiden muß, wie mit der Anfrage zu verfahren ist.

Oft findet man in Formularen einen „Reset“-Knopf. Wird er gedrückt, so wird das Formular wieder in den Ausgangszustand gebracht. Dem Attribut TYPE wird dazu der Wert RESET zugewiesen, mit VALUE kann die Aufschrift für den Knopf bestimmt werden. Da die Fälle, in denen der „Reset“-Knopf wirklich benötigt wird, relativ selten sind, sollte er nur in diesen Ausnahmefällen verwendet werden. RESET

Neben den eben beschriebenen Elementen, die alle mit der Marke <INPUT> realisiert werden, gibt es noch zwei weitere.

Für die Eingabe eines umfangreicheren Textes wurde die paarweise auftretende Marke <TEXTAREA> eingeführt. Sie besitzt neben dem Attribut NAME noch die Attribute ROWS <TEXTAREA> ROWS

COLS und COLS. Dabei werden lediglich die Maße des Festers, das der Benutzer sieht, festgelegt. Es gibt keine Möglichkeit, die Größe des eingegebenen Textes zu beschränken. Schreibt der Benutzer über den Rand des Fensters hinaus, so erzeugt der *Client* „Scrollbars“ und läßt den Benutzer nach Lust und Laune weiterschreiben.

<SELECT> Ähnliche Eigenschaften wie ein Schalter bzw. Umschalter bietet die Marke <SELECT>. Von <SELECT> und </SELECT> wird ein Menü eingeschlossen, dessen Elemente die Marken <OPTION> bilden, Text allein darf darin nicht vorkommen.

Die Marke <SELECT> hat neben NAME die Attribute SIZE und MULTIPLE.

SIZE Mit SIZE wird angegeben, wieviele Elemente des Menüs dargestellt werden sollen. Am häufigsten ist der Wert 1. Dabei wird ein Menüpunkt dargestellt. Der Rest wird erst dann ausgepackt, wenn der Benutzer diesen Menüpunkt mit der Maus anwählt. Ist
MULTIPLE zusätzlich MULTIPLE angegeben, so kann der Benutzer mehrere Menüpunkte gleichzeitig auswählen.

<OPTION> Die einzelnen Menüpunkte werden durch die Marke <OPTION> eingeleitet, danach folgt der Text, der zur Auswahl dargestellt werden soll. Mit dem Attribut VALUE kann angegeben werden, welchen Wert dieser Menüpunkt zum *Server* schicken soll, falls er ausgewählt wird. Beim Fehlen dieses Attributs wird der Text des Menüpunktes verwendet. Das Attribut VALUE sollte immer dann verwendet werden, wenn der Text aus mehreren Wörtern besteht, diakritische Zeichen und spezielle Buchstaben (č, ž, å, œ, ł...), Umlaute oder andere Sonderzeichen enthält.

Beispiel.

```
1  <FORM ACTION="mailto:Peter.Leibner@pd.siemens.de"
2     METHOD="POST" ENCTYPE="text/plain">
3
4  <TABLE BORDER=0 CELLSPACING=10>
5
6  <TR ALIGN=CENTER><TD COLSPAN=3>
7     <H1>Bewerbung f&uuml;r den Au&szlig;endienst</H1></TD></TR>
8
9  <TR ALIGN=CENTER><TD COLSPAN=3>Name
10    <INPUT TYPE=TEXT NAME="Name"></TD></TR>
11
12 <TR VALIGN=TOP>
13 <TD>Geschlecht<BR>
14       
15    <INPUT TYPE=RADIO NAME="Geschlecht" VALUE="Mann" CHECKED>
16       Mann<BR>
17       
18    <INPUT TYPE=RADIO NAME="Geschlecht" VALUE="Frau"> Frau</TD>
19 <TD ALIGN=CENTER>Arbeitsstelle in<BR>
20    <SELECT NAME="Arbeitsstelle" SIZE=1>
21    <OPTION VALUE="Muenchen" SELECTED>M&uuml;nchen
22    <OPTION>Augsburg
23    <OPTION VALUE="Sonst">Au&szlig;erhalb Bayerns
```

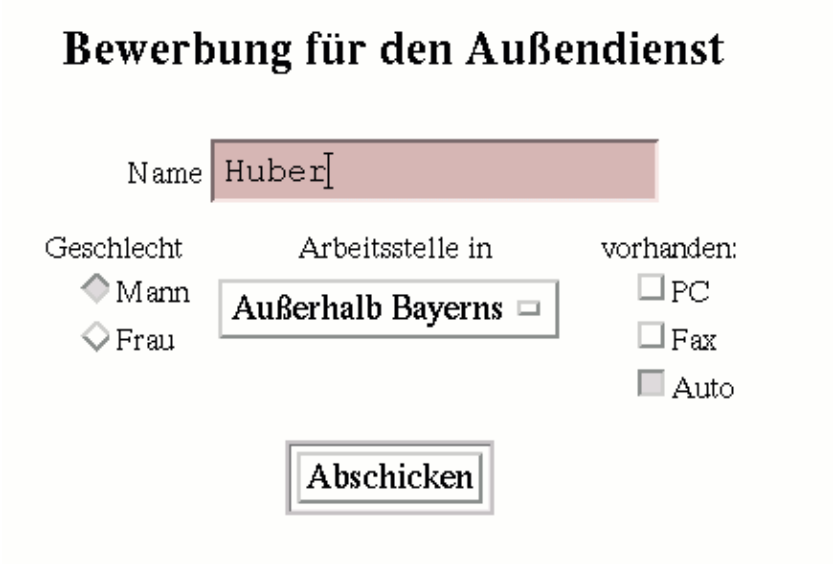


Abb. 2.15: Bildschirmausgabe, Bsp.2.3.2.8

```
24      </SELECT></TD>
25   <TD>vorhanden:<BR>
26         
27      <INPUT TYPE=CHECKBOX NAME="vorhanden" VALUE="PC"> PC<BR>
28         
29      <INPUT TYPE=CHECKBOX NAME="vorhanden" VALUE="Fax"> Fax<BR>
30         
31      <INPUT TYPE=CHECKBOX NAME="vorhanden" VALUE="Auto" CHECKED>
32         Auto
33      </TD></TR>
34   <P>
35   <TR ALIGN=CENTER>
36   <TD COLSPAN=3><INPUT TYPE=SUBMIT VALUE="Abschicken">
37   </TD></TR>
38   </TABLE>
39   </FORM>
```

Ergebnis → Abb.2.15.

Das Formular wird mit Hilfe einer Tabelle zusammengestellt (Zeilen 4 *und* 38*). Die drei Spalten werden nach den oberen Zeilen ausgerichtet (Zeile* 12*). Bei dem Umschalter (Zeilen* 15–18*) wird* `Mann` *vorgewählt. Bei den Schaltern ist es* `Auto` *(Zeile* 31*). Im Menü wird der Menüpunt* `München` *(Zeile* 21*) angezeigt.*

Wird das Formular wie in Abb.2.15 ausgefüllt und im Menü `Außerhalb Bayerns` *ausgewählt, so erhält* `Peter.Leibner@pd.siemens.de` *eine* e-mail *mit folgendem Inhalt:*

```
Name=Huber
Geschlecht=Mann
Arbeitsstelle=Sonst
vorhanden=Auto
```

2.3.2.9 Bilder mit Klinke

Das klassische Bild mit Klinke (→ Fußnote S.111) basiert auf der Auswertung der angeklickten Koordinaten durch den *Server*.

Beispiel. *Die Angaben*

```
<A HREF="/cgi-bin/imagemap/img/bild.map"><IMG
   SRC="bild.gif" ALT="" ISMAP BORDER="0"></A>
```

erzeugen eine URL, wie in Beispiel 2.18 gezeigt. Das Auswerteprogramm ist **imagemap***, das Bestandteil der WWW-*Server *von NCSA und Apache ist. Die Datei mit der Definition der Koordinaten findet der* Server *unter der URL* `/img/bild.map`.

`ISMAP` Der *Client* erkennt anhand des Attributs `ISMAP`, daß es sich bei dem Bild um ein *clickable image* handelt. Klickt der Benutzer einen aktiven Bereich des Bildes an, so wird eine HTTP-Anfrage erzeugt, deren URL sich aus der Stamm-URL, dem Weg aus `HREF` und dem Anhang `?`*x,y* zusammensetzt. Die Koordinaten entsprechen dabei dem angeklickten Punkt im Bild in Pixeln.

Auf die Anfrage hin startet der *Server* das CGI-Programm (z.B. **imagemap**) und übergibt ihm in den Umgebungsvariablen `PATH_INFO` und `PATH_TRANSLATED` (→ Bsp.2.18) den Weg zur Datei mit der Definition der aktiven Zonen. Aus `QUERY_STRING` erfährt der *Server* den Ort, den er auswerten soll. Als Antwort schickt er dann die Statusangabe `301` (`Moved Permanently`) und im Kopf die Angabe `Location` mit der zugehörigen URL. Der *Client* fordert diese Seite an und stellt sie dem Benutzer dar.

Die Syntax der Datei, welche die aktiven Gebiete definiert, hängt vom verwendeten *Server* und damit dem verarbeitenden CGI-Programm ab. Die für das CGI-Programm **imagemap** verfügbaren Kommandos sind in Tabelle 2.19 zusammengefaßt.

Kommando	Bedeutung
`default` *URL*	implizite URL; wird verwendet, wenn der Benutzer einen Punkt außerhalb der aktiven Gebiete anklickt
`circle` *URL* x_1`,`y_1 x_2`,`y_2	Kreis mit Mittelpunkt in (x_1, y_1) und einem Punkt (x_2, y_2) auf dem Kreis
`rectangle` *URL* x_1`,`y_1 x_2`,`y_2	Rechteck mit linkem oberen Punkt (x_1, y_1) und rechtem unteren Punkt (x_2, y_2) (Abkürzung: `rect`)
`poly` *URL* x_1`,`y_1 x_2`,`y_2 ... x_n`,`y_n	Polygonzug mit den angegebenen Punkten, falls der letzte nicht mit dem ersten übereinstimmt, wird der Polygonzug automatisch zwischen letztem und ersten Punkt geschlossen
`point` *URL* x`,`y	wirkt nur, wenn der Benutzer einen Punkt auswählt, der außerhalb der definierten Gebiete liegt (da nicht überall verfügbar, sollte `point` nicht verwendet werden)

Tab. 2.19: Kommandos zur Angabe aktiver Gebiete bei Verwendung von **imagemap**

Die klassische Form der Bilder mit Klinke wird von allen graphikfähigen *Clients* unterstützt. Sie ist jedoch aufgrund des Aufwands, der sowohl auf Seiten des *Clients* als auch des *Servers* getrieben werden muß, sehr uneffektiv.

Es gibt jedoch noch andere Nachteile. Da der *Client* mit der Definition der Gebiete nicht in Berührung kommt, kann er dem Benutzer auch nicht anzeigen, über welchem Gebiet sich die Maus gerade befindet und welche Seite nach dem Klicken geladen wird. Aus dem gleichen Grund ist das Bild für einen nichtgraphikfähigen *Client* überhaupt nicht zu gebrauchen. Wird die Seite außerhalb des *Internets*, z.B. zu Dokumentationszwecken oder als *On-Line*-Hilfe verwendet, so ist sie nutzlos, da zur Auswertung der angeklickten Gebiete ein *Server* benötigt wird.

Aus diesen Gründen wurde in HTML 3.2 eine neue Möglichkeit zur Auswertung der angeklickten Koordinaten geschaffen. Dazu wurden zwei neue *Tags*, nämlich `<MAP>` und `<AREA>`, eingeführt.

Mit der paarweise auftretenden Marke `<MAP>` wird der Bereich angegeben, in dem die aktiven Gebiete des Bildes definiert werden. Der Rumpf darf dabei nur `<AREA>`-Marken enthalten. `<MAP>`

Das einzige Attribut von `<MAP>` ist `NAME`. Es spielt hier eine ähnliche Rolle wie bei der Marke `<A>` (→ S.86). `NAME`

Beispiel. *In Abwandlung von Beispiel 2.3.2.9 erhält man jetzt nur noch*

```
<IMG SRC="bild.gif" ALT="" BORDER="0" USEMAP="#gebiete">
```

Eine Marke `<A>` *ist nicht mehr notwendig. Statt* `ISMAP` *wird das Attribut* `USEMAP` *verwendet, das die URL für die Definition der aktiven Gebiete zugewiesen bekommt. Steht die Definition auf der selben Seite wie das Bild, so ist dazu lediglich* #abschnitt *anzugeben.* `USEMAP`

```
<MAP NAME="gebiete">
<AREA...
  ⋮
</MAP>
```

Mit der Marke `<AREA>` werden die Gebiete definiert. Jede `<AREA>` steht jeweils für ein aktives Gebiet. Die Form des Gebietes wird durch das Attribut `SHAPE` bestimmt (→ Tab.2.20). `<AREA>` `SHAPE`

`SHAPE=`	`COORDS=`	Bedeutung
`RECT`	x_1, y_1, x_2, y_2	linke obere, rechte untere, Ecke (von links und oben gerechnet, in Pixeln)
`CIRCLE`	x_0, y_0, r	Mittelpunkt (x_0, y_0), Radius r
`POLYGON`	$x_1, y_1, x_2, y_2, \ldots, x_n, y_n$	Ecken des Polygonzuges

Tab. 2.20: Zusammenhang zwischen `SHAPE` und `COORDS`

Die Ausmaße des Gebietes werden durch das Attribut `COORDS` festgelegt. `COORDS`

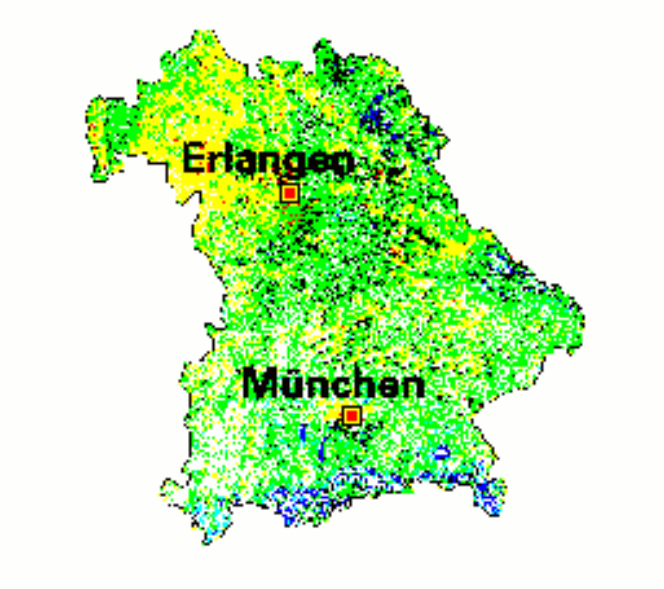


Abb. 2.16: Bildschirmausgabe, Bsp.2.20

Die URL der Seite, die nach Auswahl eines bestimmten Gebietes geladen werden soll,
`HREF` wird wieder über `HREF` definiert.

`NOHREF` Wird statt `HREF` das Attribut `NOHREF` angegeben, so wird beim Anklicken des so definierten Gebietes keine neue Seite geladen.

Beispiel.

```
1   <IMG SRC="bayern.gif" ALT="" USEMAP="#bayern" BORDER="0">
2
3   <MAP NAME="bayern">
4   <AREA SHAPE=CIRCLE
5   COORDS="80,60,20"
6   HREF="http://www.erlangen.de" ALT="Erlangen">
7
8   <AREA SHAPE=RECT
9   COORDS="90,140,120,160"
10  HREF="http://www.muenchen.de" ALT="M&uuml;nchen">
11
12  <AREA SHAPE=DEFAULT
13  HREF="http://www.entry.de/karte_Bayern.html" ALT="Bayernkarte">
14  </MAP>
```

Ergebnis → Abb.2.16.

Da ältere Clients *mit dieser Definition Schwierigkeiten haben könnten, ist es erlaubt, beide Vorgehensweisen (Auswertung durch den* Server *oder durch den* Client*) zu kombinieren.*

```
1   <A HREF="/cgi-bin/imagemap/img/bayern.map"><IMG
```



```
2      SRC="bayern.gif" ALT="" ISMAP USEMAP="#bayern" BORDER="0"></A>
3
4   <MAP NAME="bayern">
5   <AREA SHAPE=CIRCLE
6   COORDS="80,60,20"
7   HREF="http://www.erlangen.de" ALT="Erlangen"
8   onMouseOver="window.status='Homepage der Stadt Erlangen'
9      return true">
10
11  <AREA SHAPE=RECT
12  COORDS="90,140,120,160"
13  HREF="http://www.muenchen.de" ALT="M&uuml;nchen"
14  onMouseOver="window.status='Homepage der Stadt M\374nchen'
15     return true">
16
17  <AREA SHAPE=DEFAULT
18  HREF="http://www.entry.de/karte_Bayern.html" ALT="Bayernkarte"
19  onMouseOver="window.status='Bayern-Karte bei http://www.entry.de/'
20     return true">
21  </MAP>
```

Dies geschieht dadurch, daß der Marke `<IMG>` *zusätzlich das Attribut* `ISMAP` *hinzugefügt und die Marke von* `<A>` *eingeschlossen wird.* `<A>` *enthält dabei wieder den Verweis auf das CGI-Programm auf dem* Server *(Zeilen* 1–2*).*

Trifft auf so eine Seite ein Client*, der die lokale Auswertung der Verweise beherrscht, so hat er dem Attribut* `USEMAP` *Vorzug zu geben, trifft auf sie ein Vertreter der älteren Generation, so ignoriert er das für ihn unbekannte Attribut, findet dafür aber* `ISMAP` *und geht so vor, wie oben beschrieben.*

In den Zeilen 8*,* 9*,* 14*,* 15*,* 19 *und* 20 *sind Kommandos aus JavaScript verwendet worden. Diese führen dazu, daß die Angaben* `Homepage der Stadt Erlangen`*, usw., immer dann auf dem unteren Rand des Netscape-Rahmens erscheinen, wenn die Maus sich über dem jeweiligen aktiven Gebiet befindet.*

Da die meisten *Clients* inzwischen die Auswertung der Bilder mit Klinke beherrschen, sollte von der serverseitigen Auswertung Abstand genommen werden. Für *Clients*, die dies nicht können, müssen alternativ Verweise in Textform angeboten werden.

2.3.2.10 Hierarchische Stildefinition

Wie bereits erwähnt (→ S.78), bietet HTML Mittel und Wege die logische, nicht aber die typographische Struktur eines Dokumentes zu definieren.

Eine logische Strukturierung fordern z.B. automatisierte Suchwerkzeuge, die typographische kommt den ästhetischen Wünschen der Benutzer entgegen. Mit Hilfe der *Cascading Style Sheets* (CSS) wird versucht, beide Lager zufriedenzustellen. Sie erlauben die Trennung der Definition von Aussehen und Struktur eines Dokumentes. CSS

Einbindung der Stile

Die Definition der Stile kann direkt im HTML-Dokument erfolgen oder aus externen Dateien hinzugeladen werden. Es wird empfohlen, die zweite Möglichkeit zu verwenden.

<LINK> Steht die Definition in einer eigenen Datei, so wird auf sie im Kopfteil der HTML-Datei durch die Marke <LINK> verwiesen.

Beispiel.

```
<LINK REL="StyleSheet" HREF="/css/mono.css" TYPE="text/css" TITLE="BW-Monitore">
```

REL TYPE *Neben den notwendigen Attributen* REL *(gibt an, in welcher* REL*ation das bei* HREF *angegebene Dokument zum aktuellen steht) und* HREF *sind* TYPE *und* TITLE *optional.* TYPE *gibt den* Content-Type *des an den* Client *übergebenen Dokumentes an.*

Erfolgt die Definition in einer eigenen Datei, so sollte diese ausschließlich Stildefinitionen enthalten und mit dem Anhang **.css** (Empfehlung) enden. Kommentare in diesen Dateien haben die Form

```
/*
Kommentar...
*/
```

<STYLE> Mit Hilfe der Marke <STYLE> kann die Definition des Stils direkt im Kopf der HTML-Datei angegeben werden.

Beispiel.

```
<STYLE TYPE="text/css">
<!--
BODY { background-color: #FFFFFF }
-->
</STYLE>
```

Damit Clients*, die CSS nicht unterstützen, die Definition der Stile ignorieren können, müssen diese als HTML-Kommentar angegeben werden.* Clients*, die CSS verstehen, sollten diese Konstruktion dennoch richtig interpretieren.*

Beide angegebenen Arten der Integration von Stilen können auch kombiniert werden. Mit Hilfe von <LINK> kann z.B. ein gemeinsamer Stil für alle Dokumente, mit <STYLE> zusätzlich eine lokale Modifikation angegeben werden.

STYLE Sehr unschön, obwohl nicht verboten, ist es, bei einer Marke das Attribut STYLE zu verwenden. Dadurch wird die Grundidee, nämlich Struktur und Aussehen zu trennen, ad absurdum geführt.

Beispiel.

```
<P STYLE="text-align: right; color: blue">
```

Der Text des Absatzes wird rechtsbündig angeordnet und in Blau dargestellt.

Sollen bestimmte Regeln für einen Teil der HTML-Seite gelten und ist dieser Teil durch *Tags* eingeschlossen, so können den Marken die Attribute CLASS bzw. ID (s.u.) hinzugefügt werden. Ist der Text nicht Rumpf irgendeiner Marke, so können dazu die Marken

<DIV> <DIV> und <SPAN> verwendet werden.

Beispiel.

```
<STYLE><!--
.warnung { margin-right: 25%; margin-left: 25%;
background: red; color: white }
--></STYLE>
...
<DIV CLASS=warnung>
<H3>Vorsicht!</H3>
Vor und hinter dem Rumpf der Marke &lt;<TT>DIV</TT>&gt; erfolgt
ein Zeilenumbruch!
</DIV>
```

Die Marke `<SPAN>` tritt in der Regel als Teil des Textes auf (so wie z.B. `<EM>`). `<SPAN>`

Beispiel.

```
<STYLE><!--
.stadt { background: #CCCCCC; font-style: italic }
--></STYLE>
...
Die Hauptstadt der BRD ist
<SPAN CLASS=stadt>Berlin</SPAN>.
```

Das Wort Berlin wird durch diese Definition kursiv auf grauem Hintergrund dargestellt.

Deklaration der Stile

Die Grundlage für Stildeklarationen bilden Regeln der Form

selektor$_1$[`,`*selektor*$_2$[`,`...]] `{` *deklaration*$_1$[`;`*deklaration*$_2$[`;`...]] `}`

Dabei ist

selektor ein beliebiges HTML-Element und
deklaration hat die Form *eigenschaft*`:`*wert* [`!important`]

Die *eigenschaften* spezifizieren die Schrift, die Form des Textes (Ausrichtung, Einrückungen, usw.), die Farbe (Schrift, Hintergrund), die Form von vordefinierten Blöcken (Boxen), die Klassifikation und die Positionierung eines Objektes.

Der *selektor* kann wie folgt spezifiziert werden:

Kontext: Soll die Deklaration nur in einem bestimmten Zusammenhang gelten, so kann der Selektor durch die Angabe der Reihenfolge der Marken deklariert werden, in welcher sie den geforderten Kontext bilden. Die Marken werden dabei ohne `<` und `>` angegeben. Der einfachste Selektor ist eine einzige oder eine Gruppe von Marken.

Beispiel.

```
P { text-indent: 10% }
```

Rückt die erste Zeile von jedem Absatz um 10% der aktuellen Zeilenlänge ein.

```
H1, H2, H3, H4, H5, H6 { color: blue; font-family: sans-serif }
```

Stellt für Überschriften aller Stufen den Schrifttyp Sans-Serif und die Farbe Blau ein.

Mehrere Marken zusammen bilden einen Kontext.
Beispiel.

```
BLOCKQUOTE A { background-color: #FFFF99 }
```

Der Client *erzeugt aus der Folge der Marken im HTML-Dokument einen Kontext. Entspricht er dem des Selektors, so stellt er die geforderten Eigenschaften ein. Dabei kommt es nicht darauf an, daß* `<A>` *unmittelbar* `<BLOCKQUOTE>` *folgt. Die Reihenfolge*

`HTML — BODY — BLOCKQUOTE — STRONG — A`

entspricht ebenfalls der Regel und die von `<BLOCKQUOTE>` *eingeschlossenen Verweise erscheinen daher in der spezifizierten Farbe Gelb.*

Klasse: Die Deklarationen aus den Beispielen 2.3.2.10 und 2.3.2.10 gelten immer für die ganze HTML-Seite. Durch die Einführung von Klassen wurde die Möglichkeit geschaffen, Seitenabschnitten spezielle Eigenschaften zuzuordnen.
Beispiel. *Soll z.B. ein bestimmter Absatz, der eine Warnung enthält, in Rot dargestellt werden, so lautet seine Deklaration*

```
P.warnung { color: #FF0000 }
```

Die Klasse wird der Marke `<P>` *dann mit Hilfe des Attributs* `CLASS` *zugewiesen.*

```
<P CLASS="warnung">Dies sollten Sie bleiben lassen!</P>
```

`CLASS` *Der Selektor kann auch allein nur durch die Klasse angegeben werden.*

```
.warnung { color: #FF0000 }
```

Die Zugehörigkeit zu einer bestimmten Klasse wird dann auch auf die nachfolgenden Marken einer Umgebung vererbt.

```
<BLOCKQUOTE CLASS="warnung">
Die Warnung gilt f&uuml;r alle
<A HREF="http://www.microsoft.com/ie/default.asp">
Microsoft</A>-Seiten
und sollte <STRONG>beachtet</STRONG> werden!
</BLOCKQUOTE>
```

Die Marken `<A>` *und* `<STRONG>` *innerhalb von* `<BLOCKQUOTE>` *gehören folglich auch der Klasse* `warnung` *an und erben daher ihre Eigenschaften.*

Identifikator: Eine Deklaration, die nur für einen bestimmten Abschnitt gelten soll, kann auch mit Hilfe von Identifikatoren erfolgen. Der Nachteil von Identifikatoren ist jedoch, daß pro HTML-Seite nur eine Marke mit einem speziellen Identifikator versehen werden darf, wogegen gleiche Klassen unterschiedlichen Marken vergeben werden können. Von der Verwendung von Identifikatoren wird daher abgeraten.

Beispiel.

```
#unterschrift { text-align: right; font-style: italic }
```

Der Identifikator `unterschrift` *kann z.B. zusammen mit der Marke* `<P>` *verwendet werden.*

```
<P ID="unterschrift">Peter Leibner</P>
```
ID

Die Unterschrift wird rechtsbündig und kursiv gesetzt.

Pseudoklassen: Für die Marke `<A>` wurden drei Pseudoklassen eingeführt:

```
A:link { color: wert }
A:visited { color: wert }
A:active { color: wert }
```

Sie werden vom *Tag* durch einen Doppelpunkt getrennt angegeben. Mit ihrer Hilfe können Farben für gerade angeklickte (`active`), besuchte (`visited`) und im Text farblich hervorzuhebende Verweise definiert werden. Den gleichen Effekt erzielt man mit den Attributen `LINK`, `VLINK` und `ALINK` der Marke `<BODY>`.

LINK
VLINK
ALINK

Pseudoelemente: Pseudoelemente wurden bisher nur zwei und nur für die Marke `<P>` eingeführt.

Beispiel.

```
P.warnung { color: #FF0000 }
P.warnung:first-letter { font-size: 200%; float: left }
P.warnung:first-line { font-weight: bolder }
```

Warnungen (`<P CLASS="warnung">...</P>`*) fangen jetzt mit einem Initial und einer fett gedruckten ersten Zeile an.*

Die Angabe aller Möglichkeiten für die *eigenschaften* und *werte* in den Stildeklarationen würde hier zu weit führen. In den Beispielen wurden unter anderem `text-align`, `color`, `font-size` und `font-family`, verwendet. Für jede dieser Eigenschaften gibt es wiederum eine Vielzahl von *werten*, wie z.B. `center`, #FF0000 ≡ red ≡ `#f00` ≡ `rgb(255,0,0)` ≡ `rgb(100%,0,0)`, ±*n einheiten* (→ Tab.2.21) oder ±*n*`%` bzw. `monospace`.

einheiten	Bedeutung
`px`	Pixel (Bildpunkt)
`pt`	*point* (typographischer Punkt, `1pt` ≡ `1/72in`)
`pi`	*pica* (`1pi` ≡ `12pt`)
`in`	*inch* (Zoll, `1in` ≡ `2.54cm`)
`cm`	*centimeter*
`mm`	*millimeter*
`en`	Höhe von Kleinbuchstaben in der gewählten Schrift (entspricht der Höhe von x)
`em`	Höhe von Großbuchstaben in der gewählten Schrift (entspricht der Höhe von M)

Tab. 2.21: Absolute Längeneinheiten

Beispiel.

```
1   <HTML>
2   <!-- Datei beisp11.html -->
3   <HEAD>
4   <TITLE>HTML in Beispielen - CSS</TITLE>
5   <LINK REL=StyleSheet HREF="beisp11.css" TYPE="text/css">
6   </HEAD>
7   <BODY>
8   <H1 CLASS="ueberschrift">Cascaded Style Sheets</H1>
9
10  <P>Mit Hilfe der
11  <SPAN CLASS="schluesselwort">Cascaded Style Sheets</SPAN>
12  kann das Aussehen der HTML-Seite beeinflu&szlig;t werden.
13  Dies geschieht auf rein logischer Ebene.</P>
14  <P CLASS="warnung">
15  <STRONG>Vorsicht!</STRONG><BR>
16  Zur Zeit werden die CSS nur von wenigen <EM>Browsern</EM>
17  verstanden.</P>
18
19  <P>Der vorhergehende Absatz wurde in der Datei
20  <B>beisp11.css</B> als selbst&auml;ndige Klasse
21  definiert. Dies bewirkte alle Ver&auml;nderungen
22  in seiner Formatierung.</P>
23
24  <P CLASS="unterschrift">
25  <A HREF="mailto:Peter.Leibner@pd.siemens.de">Dr. Leibner</A>
26  </P>
27  </BODY>
28  </HTML>
```

Ergebnis → Abb.2.17.

Die Stildatei **beisp11.css** *wird in Zeile* 5 *mittels* `<LINK>` *eingebunden.*

Sie hat folgenden Inhalt:

```
1   BODY { background-color: #ADD8E6 }
2
3   P { margin-left: 25% }
4
5   .ueberschrift { text-align: center; font-style: italic }
6
7   .unterschrift { text-align: right; font-family: monospace }
8   .schluesselwort { background-color: #FFFF99; font-style: italic }
9
10  .warnung { color: white;
```

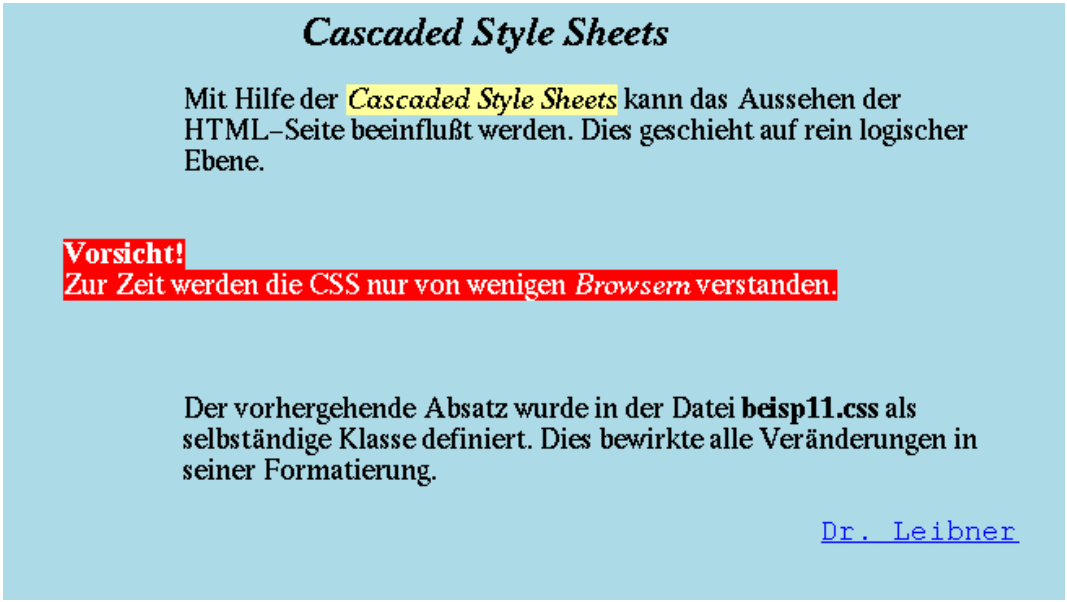


Abb. 2.17: Bildschirmausgabe, Bsp.2.21

```
11              background-color: red;
12              margin-top: 1em; margin-right: 4em;
13              margin-bottom: 2em; margin-left: 4em }
```

Die Hintergrundfarbe für die ganze Seite wird in Zeile 1 *auf* `lightblue` *eingestellt. Alle Absätze, die nicht durch einen Identifikator oder eine Klasse definiert werden, werden um 25% eingerückt (Zeile* 3*). Ist ein Identifikator oder eine Klasse spezifiziert (Zeilen* 14 *und* 24 *in der Datei* **beisp11.html***), so hat diese Angabe Vorrang.*

Die Hintergrundfarbe für die in der Datei **beisp11.html** *(Zeile* 11*) hervorgehobene Textpassage* Cascaded Style Sheets *wird in Zeile* 8 *auf Gelb gesetzt.*

2.3.2.11 JavaScript und Java

Obwohl beide Programmiersprachen ähnliche Namen haben, so unterscheiden sie sich doch entschieden in ihren Eigenschaften und in ihrer Verwendung. Die Firma Netscape hat durch die Umbenennung von LiveScript in JavaScript versucht, die Popularität von Java für ihr Produkt auszunutzen. Dennoch wurde JavaScript lange Zeit nur von Netscape unterstützt. Jetzt sieht es so aus, als ob es auch bei anderen *Browser*-Anbietern Anklang finden würde.

Zwei Dinge haben beide Sprachen gemeinsam — sie hängen mit dem WWW zusammen und sie erinnern in ihrer Syntax an die Programmiersprache C.

JavaScript

Der Unterschied zwischen einem Skript und einem klassischen Programm besteht darin, daß das Skript interpretiert wird (es ist nicht notwendig es zu übersetzen) und daß seine Zugriffsmöglichkeiten auf Systemresourcen eingeschränkt sind.

Das Skript hat nichts mit HTML zu tun. Es wird in eine HTML-Seite mit Hilfe der
<SCRIPT> Marke <SCRIPT>...</SCRIPT> eingebunden.

Das Skript zeichnet sich auf Seiten des *Clients* durch folgende Eigenschaften aus:

- der Quelltext ist in der Regel im HTML-Dokument eingebunden,
- das Skript wird durch den *Client* interpretiert (es erfolgt also keine Übersetzung des Quellprogramms),
- die Skriptobjekte werden durch den *Browser* verwaltet,
- Verweisen wird erst zum Zeitpunkt der Interpretation nachgegangen (d.h., der *Browser* meldet einen eventuellen Fehler erst dann, wenn er die fehlerhafte Stelle erreicht hat),
- Variable werden nicht explizit deklariert,
- die Interpretation ist langsamer als die Ausführung eines übersetzten Programms und
- es gibt keinen standardisierten Referenzierungsmechanismus auf HTML-Objekte.

Die Einbindung in HTML-Seiten erfolgt normalerweise im Kopfteil.

```
<HTML>
<HEAD>
  ⋮
LANGUAGE  <SCRIPT LANGUAGE="JavaScript">
<!--
  Quellkode
//-->
</SCRIPT>
  weitere Kopfzeilen
</HEAD>
<BODY>
  HTML-Seite
</BODY>
</HTML>
```

Der Kommentar ist nicht Teil der Deklaration. *Browser*, die Skriptsprachen nicht unterstützen, ignorieren dadurch den folgenden Quellkode. Die Interpreter dagegen erlauben die Zeichenkette `<!--` am Anfang des Skripts und ignorieren den Rest der Zeile. Das Ende des HTML-Kommentars muß jedoch für den Interpreter durch einen Kommentar der Skriptsprache (`//`) versteckt werden.

Das Skript kann sowohl in einer externen Datei als auch in der HTML-Seite selbst stehen. Es muß auch nicht zwischen `<HEAD>...</HEAD>` eingebunden, sondern kann überall in der HTML-Seite definiert werden.

Wird es in einer selbständigen Datei spezifiziert (*name*.**js**), so erfolgt der Zugriff auf
SRC sie über das Attribut `SRC="`*URL*`"`. Diese Datei darf keine weiteren HTML-Elemente enthalten.

Java

Die Programme in der Programmiersprache Java werden Applets genannt. Jedes Applet muß programmiert (*name*.**java**) und in einen sogenannten *Bytecode* übersetzt werden (*name*.**class**). Dieser wird vom *Server* heruntergeladen und durch den *Client* interpretiert. Das Herunterladen geschieht über die URL, die dem Attribut `CODE` oder `CODEBASE` der Marke `<APPLET>`[51] zugewiesen wird. `CODE` enthält die relative URL bezüglich der „Homepage“ der Java-Applets, die in `CODEBASE` angegeben wird (falls die Angabe fehlt, wird die URL des Stammdokuments genommen).

Applet
`CODE`
`CODEBASE`
`<APPLET>`

Beispiel.

```
<APPLET CODEBASE="classes" CODE="javex.class" WIDTH=150 HEIGHT=150>
   <PARAM NAME="bgcolor" VALUE="000000">
   <PARAM NAME="facecolor" VALUE="FFFFFF">
   <PARAM NAME="minutecolor" VALUE="008080">
   <PARAM NAME="hourcolor" VALUE="000080">
   <PARAM NAME="textcolor" VALUE="000000">
   <PARAM NAME="casecolo" VALUE="000080">
   <PARAM NAME="trimcolor" VALUE="C0C0C0">
</APPLET>
```

Für Applets wird mit `WIDTH` *und* `HEIGHT` *ein Teil der Fensterfläche reserviert.*

Mit `<PARAM>` *werden die Werte der Parameter des Applets festgelegt. Die Werte der Attribute* `NAME` *enthalten die Namen der Parameter, der Inhalt von* `VALUE` *legt die Werte dieser Parameter fest.*

`<PARAM>`
`NAME`
`VALUE`

2.3.2.12 Tricks

Die folgenden Tricks nutzen lediglich die Eigenschaften und Regeln von HTML aus, sie sind keine Programmierfinten oder ähnliches.

Filtern von Marken (<!*MARKE*>)

Der Name eines HTML-Elements muß unmittelbar hinter < folgen und sein erstes Zeichen muß immer ein Buchstabe sein. Wird diese Regel verletzt, so stellt der *Client* die fehlerhafte Konstruktion im Ausgabetext dar, so daß Tipfehler schnell erkannt und behoben werden können.

Ist das erste Zeichen hinter der spitzen Klammer ein Ausrufezeichen, so nimmt der *Client* an, daß dies der Beginn eines Kommentars ist und ignoriert den Rest. Dieser Trick kann vorteilhaft beim *Debugging* des HTML-Dokuments verwendet werden. Er ist besonders für nicht paarweise vorkommende Marken geeignet. Das Verstecken der Beginn-Marke bei paarweisen Marken kann dazu führen, daß der nachfolgende Text oder schlimmstenfalls das ganze Dokument, falsch interpretiert werden (die verwaiste Ende-Marke stört den *Client* dagegen nicht).

[51] in HTML 4.0 wurde `<APPLET>` durch `<OBJECT>` ersetzt

Beispiel.

```
<!IMG SRC="alternativbild1.gif">
<IMG SRC="alternativbild2.gif">
<!IMG SRC="alternativbild3.gif">
<!IMG SRC="alternativbild4.gif">
<!IMG SRC="alternativbild5.gif">
```

Es wird nur das Bild `alternativbild2.gif` *verwendet, die anderen Angaben werden ignoriert.*

Filtern von Attributen (<*MARKE*...*_Attribut*="*wert*">)

Da der *Client* unbekannte Attribute ignoriert, können in einer Marke mehrere Alternativen angegeben werden. Die einfachste Möglichkeit dies zu tun, ist vor dem Namen des Attributs einen Unterstrich zu schreiben.

Beispiel.

```
<BODY _BGCOLOR="#FFFFFF"
       BGCOLOR="#DCDCDC"
      _BGCOLOR="#AFDC78"
      _BGCOLOR="#000000"
      _BGCOLOR=blue>
```

Es wird nur die Farbe `DCDCDC` *verwendet, die restlichen werden ignoriert.*

Kommentare in Marken (<*MARKE*..."*kommentar*">)

Die *Clients* ignorieren unbekannte Werte innerhalb der Marken. Es ist folglich möglich, in die Marke eine beliebige Zeichenkette einzufügen, ohne daß dies die Wirkung der Marke ändern würde.

Undefinierte Attribute in Marken (<*MARKE*...*FATTRIBUT*>)

Clients ignorieren Attribute, die sie nicht identifizieren können. Daher können in Marken Attribute verwendet werden, die nicht definiert sind. Auf diesem Prinzip baut dynamisches HTML (DHTML) auf.

Beispiel.

```
<TABLE "tabelle0" BORDER=0>
            ⋮
<TD>
     <TABLE "tabelle1" BORDER=0>
     <TR tlr0>
```

```
          <TD>...</TD>
     </TR>
     <TR tlr1>
          <TD>...</TD>
     </TR>
     </TABLE>
</TD>
          ⋮
</TABLE>
```

Die Orientierung in unübersichtlichen Tabellen kann durch das Hinzufügen von Kommentaren (`"..."`) und fremden Attributen (`...`) wesentlich verbessert werden.

2.3.2.13 HTML 4.0 und CSS2

Die Geschichte von HTML ist relativ schnell erzählt.

1989 wurde in den Forschungslabors von CERN[52] das Projekt WWW[53] gestartet

1991 wurde eine vorläufige Spezifikation von HTML veröffentlicht

1992 entstand die erste funktionsfähige Version eines *Browsers* (zeilenorientiert, ohne Graphikunterstützung)

1993 Anfang des Jahres arbeiteten weltweit bereits ca. 50 WWW-*Server*,

Mitte des Jahres wurde der erste graphische *Browser* (NCSA[54] Mosaic für X Window) fertig,

im Sommer wurde der Entwurf von HTML 2.0 vorgelegt

1994 im März fand die erste internationale Konferenz statt, die sich nur dem WWW widmete,

im Frühjahr gründete der Autor des Programms Mosaic, Marc Andreseen, das Unternehmen Mosaic Communications Corp., das kurz darauf den *Browser* Netscape vorstellt,

im September wurde die gemeinnützige Organisation WWWC (*World Wide Web Consortium*) gegründet,

Ende des Jahres übergibt CERN alle WWW-Aktivitäten an das französische Institut INRIA[55]

1995 der WWW-*Boom* beginnt,

im November erfolgt die Standardisierung von HTML 2.0 (RFC 1866),

[52] *Conseil Europèen pour la Recherche Nuclèaire*
[53] von Tim Berners-Lee
[54] *National Center for Supercomputing Applications*
[55] *Institut National de Recherche en Informatique et en Automatique*

Netscape erweitert HTML 2.0 um Tabellen und Rahmen und bezeichnet diese inoffizielle Version als HTML 3.0

1996 im Mai wird die HTML-Version 3.2 spezifiziert (aus dieser Version wurden Elemente, die bereits in HTML 3.0 vorhanden waren, wieder entfernt),

Microsoft bietet seinen ersten *Browser* kostenlos an

1997 im Juli wird HTML 4.0, eine Erweiterung von HTML 3.2 (umfangreichere Tabellen und Formulare, Rahmen, fließende Rahmen, CSS, Skripte, Objekte und Internationalisierung), vorgestellt,

im Herbst folgen dynamische Elemente → DHTML

1998 Standardisierung von HTML 5.0? (HTML 4.0, erweitert um ein dynamisches Modell und die Definition von Kanälen).

Erstes Betriebssystem mit integriertem *Browser*?

Wenn anfangs noch die *Browser*-Hersteller das Tempo vorgaben, so wird es heute eher durch die laufenden Erweiterungsvorschläge (*Recommendations*), die sowohl HTML als auch CSS betreffen, bestimmt. Dies führte letztendlich dazu, daß die auf dem Markt etablierten *Browser* mit der Entwicklung nicht Schritt halten konnten und die neu hinzukommenden HTML-Elemente und CSS-Erweiterungen daher nicht kennen und folglich ignorieren.

Um zu zeigen, wohin die Entwicklung von HTML und CSS geht, werden im folgenden einige neue Vorschläge vorgestellt. Sie sollen lediglich als Beispiel dienen, eine vollständige Beschreibung von HTML 4.0 und CSS2 ergäbe ausreichend Stoff für ein ganzes Buch.

HTML 4.0

Als wichtigste Neuerung gegenüber Version 3.2 kann die Unterstützung von CSS bezeichnet werden. Die Einführung der oben angegebenen CSS-Marken wurde zwar in HTML 3.2 bereits vorbereitet, die Umsetzung erfolgte dagegen erst in HTML 4.0.

Marken wie `<CENTER>`, `<FONT>`, `<ISINDEX>`, `<BLINK>` und weitere wurden als nicht empfehlenswert deklariert. Ihre Einführung in HTML 3.2 erweckte bei vielen Benutzern Bedenken und die heutige Empfehlung des WWW-Konsortiums, sie nicht weiter zu benutzen, läßt die Frage aufkommen, was das WWW-Konsortium mit dieser Politik eigentlich verfolgt.

Der bereits in RFC 1942 niedergeschriebene Vorschlag zur erweiterten Definition von Tabellen wurde in HTML 4.0 übernommen.

Eine wichtige Neuerung stellt die Internationalisierung dar. Als Zeichensatz wurde UCS (*Universal Character Set*) nach ISO 10646[56] gewählt. Da UCS ein breites Spektrum von Sprachen abdeckt, wird angenommen, daß HTML-Autoren in der Regel nur eine Untermenge von UCS benötigen werden. Diese wird dann im HTML-Kopf angegeben. Falls der *Server* in der Lage ist, die richtigen Angaben aus dem Dokument zu ermitteln, so geschieht dies automatisch, falls nicht, so hat der Benutzer die Möglichkeit es mit
`<META>` Hilfe der Marke `<META>` selbst zu tun. Die erlaubten Werte für `charset` stehen in RFC 2045.

[56] entspricht UNICODE 2.0

Beispiel.

```
<META HTTP-EQUIV="Content-Type"
   CONTENT="text/html; charset=iso-8859-2">
```

HTTP-EQUIV
CONTENT

Eine zweite Möglichkeit bietet das Attribut LANG, das in jeder beliebigen Marke vorkommen darf.

LANG

Die Marke <APPLET> wurde durch <OBJECT> ersetzt. Mit Hilfe von <OBJECT> kann nun alles mögliche in die HTML-Seiten eingebunden werden. Durch die Marke <PARAM> (→ Bsp.2.3.2.11) in ihrem Rumpf kann zudem die Funktion der eingebundenen Objekte beeinflußt werden.

<OBJECT>

Kann der *Client* ein bestimmtes Objekt nicht darstellen, so können ihm durch die Schachtelung von <OBJECT>-Umgebungen Alternativen zur Auswahl angeboten werden.

Das Attribut LANGUAGE der Marke <SCRIPT> sollte nicht länger hergenommen werden. Es wurde durch TYPE zusammen mit den Sprachtypen (→ MIME), text/javascript, text/vbscript bzw. text/tcl, ersetzt.

TYPE

Rahmen halten Einzug in die offizielle Version von HTML. HTML 4.0 enthält aber keine revolutionär neue Gedanken, es konserviert lediglich den aktuellen Stand.

Im Bereich der Formulare wurden nur Details korrigiert. Es entstanden zwei neue Eingabefelder — zum Abschicken von Dateien (<INPUT TYPE=FILE>) und ein einfacher Knopf (<INPUT TYPE=BUTTON> bzw. <BUTTON>). Jedem Eingabefeld kann mit <LABEL> eine Beschreibung zugeordnet werden. Klickt der Benutzer die Beschreibung an, so wird das zugehörige Feld aktiviert.

FILE
BUTTON
<BUTTON>
<LABEL>

Beispiel.

```
<LABEL>Name und Vorname:
<INPUT TYPE="TEXT" NAME="name">
</LABEL>
```

Mit Hilfe der Attribute FOR in <LABEL> und ID in <INPUT> kann die Beschreibung einem beliebigen Eingabefeld zugeordnet werden.

FOR
ID

Beispiel.

```
<LABEL FOR="nv">Name und Vorname:</LABEL>
                  ⋮
<INPUT TYPE="TEXT" NAME="name" ID="nv">
```

Neben den hier erwähnten neuen Marken und Attributen ist eine Vielzahl weiterer in HTML 4.0 hinzugekommen. Für den interessierten Anwender empfiehlt es sich, dazu die Spezifikation von HTML 4.0[57] durchzulesen.

CSS2

Da viele der heutigen *Browser* nicht einmal in der Lage sind, alle Elemente von CSS1 richtig auszuwerten, soll an dieser Stelle auf CSS2 nur kurz eingegangen werden. Genauere Informationen erhält man wieder vom WWW-Konsortium[58].

Durch die Angabe von ~ zwischen den Marken kann die im Beispiel 2.3.2.10 gezeigte

~

[57] http://www.w3.org/TR/REC-html40/
[58] http://www.w3.org/TR/WD-CSS2/

Kontext-Regel strenger gefaßt werden.

Beispiel.

```
A ~ EM { color: red }
```

schreibt vor, daß das Wort `hier` *in*

```
<A HREF="datei.html"><EM>hier</EM>steht es rot!</A>
```

rot geschrieben wird, in

```
<A HREF="datei.html"><B><EM>hier</EM></B>jedoch nicht!</A>
```

jedoch nicht.

Den Seiten können nun Kopf- und Fußzeilen hinzugefügt werden. In ihnen können z.B. der Titel des Dokuments, seine URL, eine Seitennumerierung und die Kapitelüberschriften spezifiziert werden.

Die erweiterten Möglichkeiten, eine Seite typographisch zu gestalten, kommen insbesondere den vielfältigen Ausgabemöglichkeiten zugute. Die Ausgabemedien werden dabei in endlose und seitengebundene eingeteilt. Zu den endlosen gehören z.B. `SCREEN`, zu den seitengebundenen `PRINT`. Weitere Medien sind in der nachfolgenden Tabelle (→ Tab.2.22) zusammengefaßt.

SCREEN
PRINT

Medium	Bedeutung
`SCREEN`	Bildschirm
`PRINT`	Drucker, bzw. Darstellung der gedruckten Form auf dem Bildschirm
`PROJECTION`	Folienausgabe
`BRAILLE`	Braille-Terminal für Blinde
`AURAL`	synthetische Sprachausgabe
`TV`	Ausgabe auf dem Fernseher
`HANDHELD`	Taschenrechner mit kleinem monochromatischem Bildschirm und niedriger Übertragungsrate
`ALL`	für alle Ausgabemedien

Tab. 2.22: Von CSS2 unterschiedene Medien

Auch in Bezug auf die Definition von Schriften hat sich einiges getan. Wird in CSS1 noch angenommen, daß alle Schriften auf dem *Client* vorhanden und durch ihren Namen eindeutig identifizierbar sind, so muß dies bei CSS2 nicht mehr der Fall sein.

Jedem *Client* steht im Normalfall eine Datenbasis mit unterschiedlichen Schriften zur Verfügung. Für CSS1 ist diese Datenbasis die einzige Quelle, die zur Verfügung steht.

Auch CSS2 benutzt diese Datenbasis als ihre primäre Quelle. Wird durch den Autor einer Stildatei eine bestimmte Schrift angefordert, so stellt der *Client* mit Hilfe eines Algorithmus (*Font Matching Algorithm*) zunächst fest, welche Schrift mit dieser übereinstimmt oder ihr am nächsten kommt. Danach lädt er sie entweder aus der lokalen Datenbasis, erzeugt sie selbst, oder holt sie aus dem WWW.

Es stehen vier Auswahlverfahren zur Verfügung:

Font Name Matching In diesem Fall wählt der *Client* eine Schrift, die den gleichen Namen hat, wie die angeforderte. Dabei müssen das Schriftbild und die Metrik nicht unbedingt übereinstimmen (z.B. bei Schriften unterschiedlicher Hersteller).

Intelligent Font Name Matching In diesem Fall benutzt der *Client* eine lokal verfügbare Schrift, die der angeforderten am nächsten kommt. Auch hier muß die Metrik nicht unbedingt die gleiche sein.

Font Synthesis In diesem Fall erzeugt der *Client* eine Schrift, die mit der angeforderten auch in Bezug auf die Metrik exakt übereinstimmt.

Download Die letzte Möglichkeit besteht darin, die Schrift über das WWW herunterzuladen. Die Vorgehensweise ist dabei die gleiche, wie beim Herunterladen von Bildern oder Applets.

Beispiel.

```
<HTML>
<HEAD>
<TITLE>Font-Test</TITLE>
<STYLE TYPE="text/css" MEDIA="SCREEN, PRINT">
@font-face { font-family: "Robson Celtic";
  src: url(http://www.firma.de/fonts/rob-celt) }
H1 { font-family: "Robson Celtic", serif }
</STYLE>
</HEAD>
<BODY>
<H1>Diese &Uuml;berschrift wird in Robson Celtic gesetzt</H1>
</BODY>
</HTML>
```

Bei CSS1 wird nach der Schrift „Robson Celtic“ gesucht. Wird sie nicht gefunden, so wird eine Serif-Schrift genommen, die dem Client *bekannt ist.*

Ein Client*, der CSS2 implementiert hat, wird zuerst die* `@font-face`*-Regeln nach einer Beschreibung der Schrift „Robson Celtic“ durchsuchen. Solch eine Regel ist vorhanden, obwohl sie nicht sehr viele Daten über die Schrift enthält. Sie gibt jedoch durch eine URL an, wo die Schrift geholt werden kann. Findet der* Client *keine* `@font-face`*-Regel, so geht er wie ein* Client *vor, der nur CSS1 implementiert hat.*

2.3.3 WWW-Server

Da die Erstellung einer WWW-Server-Statistik relativ leicht automatisiert werden kann, spiegeln die veröffentlichten Ergebnisse (→ Abb.2.18) ziemlich genau wider, auf welchen Schienen das WWW eigentlich läuft.

Die von Netcraft[59] erstellte Übersicht basiert auf der Untersuchung von über zwei Millionen (2 084 473) *Servern*, die einen *Internet*-Service anbieten. Die erste solche Untersuchung im Jahre 1995 ergab eine Anzahl von 18 957 *Internet-Servern*.

[59] `http://www.netcraft.com/survey/`

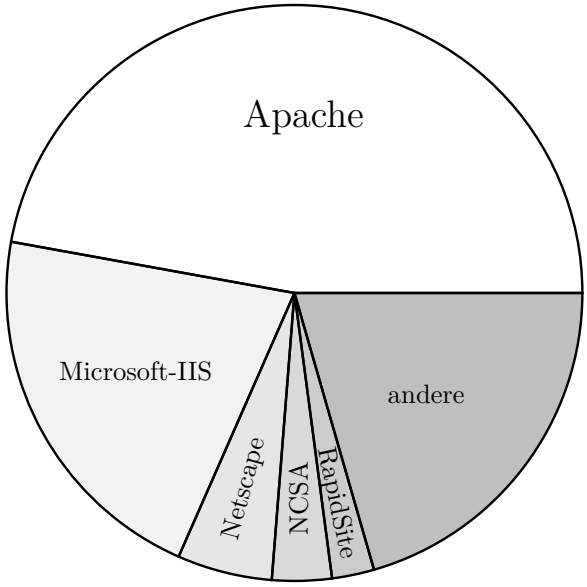


Abb. 2.18: Verteilung der WWW-Server weltweit (Stand: März 1998)

Die Statistik bestätigt die führende Rolle des Apache-Servers (46.79%), gefolgt von den *Servern* von Microsoft-IIS (21.10%), Netscape-Enterprise (5.29%), NCSA[60] (3.32%) und RapidSite (2.35%).

Apache

Da der Apache-Server eine zunehmend dominantere Rolle unter den zur Zeit verwendeten *Servern* spielt, soll er an dieser Stelle kurz besprochen werden.

Der Apache-Server benötigt folgende vier Konfigurationsdateien:

httpd.conf Die Konfigurationsdatei für das Programm **httpd**. Sie enthält Direktiven (*directives*), die den Lauf des *Servers* beeinflussen.

srm.conf Die Konfigurationsdatei für die angebotenen Dokumente.

access.conf Datei, in welcher die Zugriffsrechte der *Clients* zu einzelnen Verzeichnissen festgelegt werden. Die Zugriffrechte für ein Verzeichnis können auch in der Datei **.htaccess**, die in jedem Verzeichnis ein anderes Aussehen haben kann, bestimmt werden.

mime.types Konfigurationsdatei für MIME, in der unterschiedlichen Suffix von Dateinamen die Typen und Subtypen von MIME zugeordnet werden. Diese Datei sollte nicht verändert werden. Falls ein Typ fehlen sollte, so kann er in **srm.conf** hizugefügt werden.

Allgemein gilt, daß jede Zeile maximal eine Direktive enthalten darf. Die Zeile beginnt mit der Direktive, gefolgt von dem Wert, auf den sie eingestellt werden soll. Als Trennzeichen dienen ein oder mehrere Leer- bzw. Tabulatorzeichen. Groß- und Kleinschreibung wird nur bei Datei- und Verzeichnisnamen sowie in den URLs unterschieden.

[60] die Modifikation dieses *Servers* führte zum Apache-Server (z.B. `http://www.apache.de`), der wegen seiner freien Verfügbarkeit und aufgrund der Abwärtskompatibilität seiner Konfigurationsdateien zu denen des NCSA-Servers, diesen allmählich verdrängt

Das Wurzelverzeichnis für die Installation des *Servers* ist auf **/usr/local/apache** voreingestellt. In diesem Verzeichnis liegt das *Server*-Programm **httpd** und die Unterverzeichnisse **conf**, **logs**, **icon** und **cgi-bin**. Unter **conf** sind die vier Konfigurationsdateien untergebracht, unter **logs** liegen die Protokolldateien und **icon** enthält verschiedene *Icons* zur Darstellung von Dateien, Verzeichnissen, usw. Im Verzeichnis **cgi-bin** werden die CGI-Programme abgelegt. Das Wurzelverzeichnis kann durch das Setzen der Direktive `ServerRoot` neu eingestellt werden. Alle Wege sind relativ zu *`ServerRoot`*, sofern sie nicht mit `/` beginnen. `ServerRoot`

Wird der Inhalt einer der Konfigurationsdateien verändert, so muß dem *Server* das Signal HUP oder USR1 geschickt werden. Der *Daemon* **httpd** speichert beim Start seine PID[61] in der Datei *`ServerRoot`***/logs/httpd.pid**. Dies kann zum „Restart“ des *Servers* ausgenutzt werden.

Beispiel. *Läge der WWW-Server z.B. auf dem Rechner* `extor.firma.de`*, so könnte das erneute Einlesen der veränderten Konfigurationsdateien wie folgt aussehen:*

```
extor:~# kill -HUP `cat /usr/local/apache/logs/httpd.pid`
```

Durch das Signal TERM wird das Programm beendet.

Normalerweise läuft der **httpd**-*Daemon* kontinuierlich[62], so daß Anfragen jederzeit beantwortet werden können. Alternativ kann der *Server* durch den *Internet-Daemon* **inetd** immer nur dann gestartet werden, wenn eine Anfrage erfolgt[63]. Diese zweite Möglichkeit wird jedoch nicht empfohlen.

Im Lieferumfang des *Servers* sind Konfigurationsdateien (sie enden alle mit **-dist**), die an wenigen Stellen der aktuellen Serverumgebung angepaßt werden müssen.

2.3.3.1 Konfiguration des Server-Programms

Die Konfiguration des *Server*-Programms erfolgt in der Datei **httpd.conf**.

Beispiel. *Soll z.B. der Rechner* `extor.firma.de` *als WWW-Server eingerichtet werden, so könnte die Datei* **httpd.conf** *folgenden Inhalt haben:*

```
1  ServerType standalone
2  Port 80
3  HostnameLookups off
4  User http
5  Group www
6  BrowserMatch Mozilla/2 nokeepalive
7  ServerAdmin webmaster@www.firma.de
8  ServerRoot /usr/local/apache
9  ErrorLog logs/error_log
10 TransferLog logs/access_log
```

[61] genauer gesagt die PID des Vaterprozesses
[62] Direktive `ServerType` ist `standalone` (impliziter Wert)
[63] die Direktive `ServerType` ist dann `inetd`

```
11  PidFile logs/httpd.pid
12  ScoreBoardFile logs/apache_status
13  Timeout 300
14  KeepAlive On
15  MaxKeepAliveRequests 100
16  KeepAliveTimeout 15
17  MinSpareServers 5
18  MaxSpareServers 10
19  StartServers 5
20  MaxClients 150
21  MaxRequestsPerChild 30
```

In der Zeile 6 *wird durch* `nokeepalive` *angegeben, daß jede Anfrage einzeln beantwortet werden soll. Diese Direktive ist nötig, da es sonst mit* Browsern *der Versionsnummern 2.x von Netscape Probleme geben kann.*

`StartServers`

Wird der *Server* „standalone“ gestartet, so erzeugt er automatisch mehrere Sohnprozesse. Die Anzahl wird durch die Direktive `StartServers` (→ Z.19[64]) bestimmt. Immer, wenn eine Anfrage eintrifft, übergibt der *Server* sie zur Bearbeitung einem Sohnprozeß. Er sorgt gleichzeitig dafür, daß immer genügend Sohnprozesse zur Verfügung stehen.

`MinSS.` `MaxSS.`

Die Anzahl nichtstuender Prozesse wird durch die Direktiven `MinSpareServers` und `MaxSpareServers` eingegrenzt (→ Zn. 17, 18). Sinkt die Anzahl der freien Söhne unter `MinSpareServers`, so startet der Vaterprozeß einen neuen Sohn, steigt sie über `MaxSpareServers`, so wird einer der Söhne beendet. Dieser Mechanismus sorgt dafür, daß die Beantwortung von Anfragen beschleunigt und gleichzeitig die Rechnerauslastung minimiert wird.

`MRPCh.`

Die maximale Anzahl der Anfragen, die während seiner Lebenszeit ein Sohn beantworten darf, wird durch `MaxRequestsPerChild` (→ Z.21) angegeben. Hat ein Sohnprozeß die vorgegebene Grenze erreicht, so stirbt er.

`MaxClients`

Durch `MaxClients 150` (→ Z.20) wird die Anzahl der *Clients*, die gleichzeitig eine Anfrage stellen dürfen, auf 150 limitiert. Der Vaterprozeß läßt folglich nicht zu, daß gleichzeitig mehr als 150 Söhne laufen.

`User` `Group`

Durch `User` und `Group` werden die Gruppe und der Eigentümer der *Server*-Prozesse bestimmt. Der Eigentümer, die Gruppe und das Konto des Benuters `http` sollten nach den gleichen Regeln wie die des anonymen FTP-*Clients* eingerichtet werden (→ S.62) und zu keinem anderen Zweck dienen.

`Port`

Die implizite Portnummer des HTTP-*Servers* ist 80 (→ Z.2). Neben dieser Direktive gibt es zwei weitere, mit deren Hilfe der *Port* konfiguriert werden kann (→ `VirtualHost`). Es sind dies die Direktiven `BindAddress` und `Listen`.

`BindAddress`

Mit `BindAddress` wird dem *Server* mitgeteilt, unter welchen IP-Adressen er auf ankommende Anfragen hören soll. Dies macht nur dann Sinn, wenn der Rechner mehrere Netzschnittstellen hat. Als Werte können `*` (alle Netzschnittstellen) bzw. die IP-Adresse oder der Name der Netzschnittstelle, die abgefragt werden soll, angegeben werden.

[64] die folgenden Zeilenangaben beziehen sich alle auf das Beispiel 2.3.3.1

Mit `Listen` können weitere Portnummern angegeben werden, auf denen der *Server* bereit ist, Anfragen zu empfangen. Dies kann mit einer optionalen IP-Adresse kombiniert werden. `Listen`

Beispiel. *Soll Apache auf dem Rechner* `extor.firma.de` *auf den* Ports *80 und 8080 aller und zusätzlich auf dem* Port *8000 der inneren Netzschnittstelle arbeiten, so sind in der Datei* **httpd.conf** *die folgenden Zeilen hinzuzufügen:*

```
Listen 80
Listen 8080
Listen 192.168.33.94:8000
```

Das Wurzelverzeichnis wird durch `ServerRoot` (→ Z.8) eingestellt. In der Version 1.2.6 von Apache ist es in **httpd.conf-dist** auf `/usr/local/etc/httpd` eingestellt. Die zugehörige Dokumentation empfiehlt jedoch `/usr/local/apache`. Dieser Empfehlung wurde im Beispiel 2.3.3.1 Folge geleistet. `ServerRoot`

Die Direktive `Timeout` (→ Z.13) (in Sekunden) gibt an, wie lange der *Server* bereit ist zu warten, daß nach Herstellung einer Verbindung der *Client* eine Anfrage stellt und nach dem Empfang der Antwort diese quittiert. Reagiert der *Client* nicht in der angegeben Zeit, so beendet der *Server* die Verbindung. `Timeout`

Die aktuelle Version 1.0 von HTTP stellt für jede Anfrage/Antwort zwischen *Client* und *Server* eine neue Verbindung her (→ S.68). Bewegt sich der Benutzer im Datenbereich des *Servers*, so wird für jeden Ortswechsel eine neue Verbindung hergestellt. Dies stellt eine Verschwendung von Übertragungskapazität und -zeit dar. Die HTTP-Version 1.1 wurde daher um eine neue Eigenschaft erweitert, wodurch die Verbindung auch nach erfolgter Anfrage/Antwort aufrechterhalten bleiben kann. Apache unterstützt dies durch die Direktive `KeepAlive` (→ Z.14). Unterstützen beide Teilnehmer diese Möglichkeit, so bleibt die Verbindung auch nach Beendigung einer Interaktion erhalten. Sie wird erst dann beendet, wenn der *Client* eine Verbindung zu einem anderen *Server* herstellt oder längere Zeit inaktiv ist. `KeepAlive`

Die Zeit, die der *Server* bei bestehender Verbindung bereit ist, auf eine neue Anfrage zu warten, wird durch `KeepAliveTimeout` (→ Z.16) bestimmt. `KeepAT.`

Durch `MaxKeepAliveRequests` wird die Anzahl der Anfragen während einer bestehenden Verbindung angegeben. In Zeile 15 werden 100 Anfragen zugelassen (wird der Wert auf 0 gesetzt, so ist die Anzahl unbeschränkt). `MaxKeepAR.`

Über seine Tätigkeit führt Apache Buch. Dazu werden unterschiedliche Protokolldateien angelegt.

Der Pfad zur wichtigsten von ihnen wird durch die Direktive `TransferLog` (→ Z.10) bestimmt. Jede Anfrage eines *Clients* wird in einer eigenen Zeile dieser Datei protokolliert. Die Zeile enthält den Rechner (Name oder IP-Adresse), der die Anfrage stellte, die Identifikation des Benutzers (falls dies der *Client* gestattet, d.h., **identd** auf ihm läuft), den Benutzernamen (wird benötigt, falls der Zugriff auf ein passwortgeschütztes Dokument erfolgt), das Datum, wann die Anfrage erfolgte, die angeforderte URL, den Status, der dem *Client* zurückgeschickt wurde und die Größe der Antwort in Bytes. `TransferLog` **identd**

HostnameL.

Ob beim Protokollieren die Rechnernamen oder nur die IP-Adressen verwendet werden sollen, wird durch `HostnameLookups on` (Rechnername) bzw. `off` (IP-Adressen) festgelegt. Da die Ausgabe der Namen den Zugriff auf den *Server* verlangsamen kann, sollte diese Direktive auf `off` gestellt werden. Ist der Rechnername nicht zu ermitteln, wird automatisch die IP-Adresse verwendet.

2.3.3.2 Organisation der HTML-Dokumente

Alle Direktiven, die Organisation der HTML-Dokumente betreffend, werden in der Datei *ServerRoot* **/conf/srm.conf** getroffen.

Beispiel. *Die meisten Voreinstellungen können aus der Datei* **srm.conf-dist** *übernommen werden. Bei einigen wenigen müssen die Werte den lokalen Gegebenheiten angepaßt werden. Weitere Direktiven, wie z.B.* `Alias`*, können durch Auskommentieren der entsprechenden Zeile in der Datei* **srm.conf** *hinzugefügt werden.*

```
1  DocumentRoot /usr/local/apache/htdocs
2  UserDir public_html
3  DirectoryIndex index.html
4  FancyIndexing on
5  AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
6  AddIconByType (TXT,/icons/text.gif) text/*
7  AddIconByType (IMG,/icons/image2.gif) image/*
8  AddIconByType (SND,/icons/sound2.gif) audio/*
9  AddIconByType (VID,/icons/movie.gif) video/*
10 AddIcon /icons/binary.gif .bin .exe
11 AddIcon /icons/binhex.gif .hqx
12 AddIcon /icons/tar.gif .tar
13 AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
14 AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
15 AddIcon /icons/a.gif .ps .ai .eps
16 AddIcon /icons/layout.gif .html .shtml .htm .pdf
17 AddIcon /icons/text.gif .txt
18 AddIcon /icons/c.gif .c
19 AddIcon /icons/p.gif .pl .py
20 AddIcon /icons/f.gif .for
21 AddIcon /icons/dvi.gif .dvi
22 AddIcon /icons/uuencoded.gif .uu
23 AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
24 AddIcon /icons/tex.gif .tex
25 AddIcon /icons/bomb.gif core
26 AddIcon /icons/back.gif ..
27 AddIcon /icons/hand.right.gif README
28 AddIcon /icons/folder.gif ^^DIRECTORY^^
29 AddIcon /icons/blank.gif ^^BLANKICON^^
30 DefaultIcon /icons/unknown.gif
31 ReadmeName README
```

```
32  HeaderName HEADER
33  IndexIgnore .??* *~ *# */HEADER* */README* */RCS
34  AccessFileName .htaccess
35  DefaultType text/plain
36  AddEncoding x-compress Z
37  AddEncoding x-gzip gz
38  AddLanguage en .en
39  AddLanguage fr .fr
40  AddLanguage de .de
41  AddLanguage da .da
42  AddLanguage el .el
43  AddLanguage it .it
44  LanguagePriority en fr de
```

Die Direktive `DocumentRoot` (→ Z.1[65]) bestimmt den Pfad zum Wurzelverzeichnis der angebotenen HTML-Dokumente. Apache ergänzt den Pfad um den ermittelten Weg aus der URL der Anfrage (→ S.75) und schickt die gewünschte Seite dem *Client* zur Darstellung. `DocumentRoot`

`UserDir public_html` (→ Z.2) gibt an, daß HTML-Dokumente eines Benutzers mit einem eigenen Konto auf dem WWW-*Server* im Verzeichnis **public_html** gesucht werden sollen. Dieses Verzeichnis muß ein Unterverzeichnis des Heimatverzeichnisses des Benutzers sein. `UserDir`

Beispiel. *Hat der Benutzer* `vitali` *ein Konto auf dem Rechner* `extor.firma.de`*, so erhält ein* Client *nach Angabe der URL*

`http://www.firma.de/~vitali/score.html`

die Datei

`/home/vitali/public_html/score.html`

Die Tilde `~` *wird dabei zu* `/home/vitali` *(***$HOME***) expandiert und* `public_html` *aus* `UserDir` *eingefügt.*

Die Direktive `Alias` macht es möglich, Dokumente auch aus anderen Verzeichnissen als dem unter `DocumentRoot` angegebenen anzubieten. Das gleiche gilt für `ScriptAlias`, wobei die Angabe „Script“ bereits darauf hinweist, daß es sich dabei um CGI-Programme handeln wird. `Alias`

Beispiel. *Die Datei* **srm.conf** *auf dem* Server `extor.firma.de` *könnte z.B. folgende zusätzliche Zeilen enthalten:*

```
Alias / /usr/local/apache/htdocs
Alias /firma /usr/local/docs/firma
ScriptAlias /cgi-bin /usr/local/apache/cgi-bin
```

In der ersten Zeile wird die Wurzel des Weges aus der URL auf den Wert aus der Direktive `DocumentRoot` *abgebildet. Die URL*

[65] folgende Zeilenangaben beziehen sich auf Beispiel 2.3.3.2

`http://www.firma.de/info.html`

führt somit auf

`/usr/local/apache/htdocs/info.html`

In der zweiten Zeile wird der Weg `/firma` *nach* `/usr/local/docs/firma` *abgebildet.*

Die dritte Zeile gibt an, daß CGI-Programme, die in der URL als `/cgi-bin/···` *angegeben werden, unter* `/usr/local/apache/cgi-bin/···` *zu suchen sind.*

`DirectoryI.`

Gibt der Benutzer in der URL keine konkrete Datei sondern nur ein Verzeichnis an, so schaut der *Server* zuerst nach, ob in diesem Verzeichnis eine Datei mit einem Namen, so wie er bei `DirectoryIndex` angegeben wurde, existiert. Gibt es diese Datei, so erhält der *Client* sie als Antwort.

Gibt es solch eine Datei nicht, so hängt die Reaktion des *Servers* von der Einstellung der Direktive `Options` in der Datei **access.conf** ab. Lautet sie `Options Indexes`, so erhält der *Client* eine formatierte Ausgabe des Verzeichnisinhalts.

`FancyI.`

Bei `FancyIndexing on` (→ Z.4) werden bei der Ausgabe den unterschiedlichen Einträgen im Verzeichnis passende *Icons* zugeordnet.

`AddIcon`

Den Dateien im Verzeichnis wird über die Direktive `AddIcon` (→ Zn.10–29) ein *Icon* zugeordnet. Durch die spezielle Zeichenkette `^^DIRECTORY^^` (→ Z.28) wird ein Verzeichnis, durch `^^BLANKICON^^` (→ Z.29) eine Leerzeile, angegeben.

Die vorhandenen Einträge können auch über ihren MIME-Typ bzw. -Kodierung identifiziert werden.

`AddIconBT.`

Durch die Direktive `AddIconByType` (→ Zn.6–9) wird der MIME-Typ zugeordnet. In den Klammern steht an erster Stelle jeweils ein alternativer Text für *Clients*, die keine graphische Ausgabe gestatten (diese Form kann auch bei allen anderen `AddIcon...`-Direktiven verwendet werden).

`AddIconBE.`

Die Direktive `AddIconByEncoding` (→ Z.5) ordnet ein *Icon* in Abhängigkeit von der MIME-Kodierung zu.

`DefaultIcon`

Der Wert aus der Direktive `DefaultIcon` (→ Z.30) wird dann genommen, wenn ein Eintrag nicht zugeordnet werden kann.

`IndexIgnore`

Ignoriert werden alle Einträge[66], die (z.B. mit Hilfe von *Wildcards*) den Wert der Direktive `IndexIgnore` (→ Z.33) bilden.

`HeaderName` `ReadmeName`

Vor und hinter der Ausgabe können die Inhalte der bei `HeaderName` und `ReadmeName` angegebenen Dateien eingefügt werden.

`Redirect`

Ist eine Seite auf eine andere URL verlegt worden, so kann dies dem *Client* mittels der Direktive `Redirect` mitgeteilt werden. Wünscht der *Client* sich eine Seite, die bei `Redirect` angeführt ist, so schickt ihm der *Server* im Kopf der Antwort den Status 301 (→ S.104) und die neue URL.

Um dem *Client* im Kopf der Antwort mitteilen zu können[67], um was für eine Art von Information es sich handelt, muß der *Server* in der Lage sein, diese zu klassifizieren.

[66] z.B. Dateinamen, die mit einem Punkt beginnen
[67] `Content-Type` (→ S.46)

Beispiel. *Der Dateiname* **guide.ps.en.gz** *enthält z.B. die Information, daß es sich um eine PostScript-Datei in englischer Sprache handelt, die mit Hilfe des Programms* **gzip** *komprimiert wurde.*

Die unterschiedlichen Suffix des Dateinamens werden durch die Direktiven `AddType`, `AddLanguage` (→ Zn.38–43) und `AddEncoding` (→ Zn.36,37), bestimmt. `AddLanguage` `AddEncoding`

Die ermittelten Suffix für den Typ, die Sprache und die Kodierung sind in MIME (→ S.46) spezifiziert.

Die grundlegenden MIME-Typen werden in der Datei **mime.types** definiert. Diese Datei sollte nicht verändert werden. Möchte man einen weiteren Typ hinzufügen, so kann dies durch `AddType` in der Datei **srm.conf** geschehen. `AddType`

Beispiel. *Eine Möglichkeit, alle Dateien mit dem Suffix* **.cgi** *als CGI-Programme zu deklarieren, ist*

```
AddType application/x-httpd-cgi .cgi
```

Aus Sicherheitsgründen wird dies jedoch nicht empfohlen, besser ist es, `ScriptAlias` *zu verwenden.*

Kann aus dem Dateinamen kein Typ ermittelt werden, so wird der Wert aus der Direktive `DefaultType` (→ Z.35) verwendet. `DefaultType`

2.3.3.3 Einstellung der Zugriffsrechte

Der Zugriff auf den *Server* wird durch die Direktiven in der Datei **access.conf** bestimmt. Die Einschränkungen können aufgrund von gegebenen IP-Adressen oder Domänennamen oder über Benutzernamen und Passwörter erfolgen. Diese Einschränkungen müssen nicht für den gesamten Bereich der HTML-Verzeichnisse die gleichen sein, sondern können für jedes Verzeichnis individuell eingestellt werden.

Beim Wechsel aus *`DocumentRoot`* in ein Unterverzeichnis untersucht der *Server* in jedem Verzeichnis, das er beim Wechsel durchläuft, ob es

1. einen Eintrag in der Datei **access.conf** mit speziellen Anweisungen für dieses Verzeichnis gibt, oder ob es
2. in dem Verzeichnis eine Datei **.htaccess**[68] gibt, die solche Anweisungen enthält.

Der **Server** führt gegebenenfalls die Anweisungen zuerst aus und wechselt danach in das nächste Unterverzeichnis, wo sich der Vorgang dann wiederholt.

Beispiel. *Die mitgelieferte Datei hat folgende Zeilen auskommentiert:*

```
1  <Directory /usr/local/apache/htdocs>
2  Options Indexes FollowSymLinks
3  AllowOverride None
4  order allow,deny
```

[68] der Name der Datei wird durch die Direktive `AccessFileName` (→ **srm.conf**, Z.34) festgelegt

```
5   allow from all
6   </Directory>
7   <Directory /usr/local/apache/cgi-bin>
8   AllowOverride None
9   Options None
10  </Directory>
```

<Directory>

Der Gültigkeitsbereich für die Direktiven wird durch Befehle eingeschlossen, die an Tags *in HTML erinnern. Durch* `<Directory` verzeichnis`>` *und* `</Directory>` *wird eine Gruppe von Direktiven eingeschlossen, die für das angegebene Verzeichnis und seine Unterverzeichnisse gültig sind.*

Die wichtigsten Direktiven aus der Datei **access.conf** sind folgende:

`Options` (→ Zn.2,9[69]) schränkt die Möglichkeiten, die der *Server* in dem gegebenen Verzeichnis hat, auf die angeführten, ein. So erlaubt z.B. `Indexes` die Ausgabe des Verzeichnisinhalts und `FollowSymLinks` die Verwendung von symbolischen Verweisen, während `None` sämtliche Aktionen im Verzeichnis verbietet.

`AllowOverride` spezifiziert die Direktiven, deren Einstellung gegenüber der globalen in der Datei **access.conf** lokal in **.htaccess** geändert werden darf. Ist der Wert `None` (→ Zn.3,8), so werden die Dateien **.htaccess** ignoriert.

`order` gibt die Reihenfolge an, in der die Direktiven `deny` und `allow` gelesen werden sollen. Die zuletzt gelesene Direktive, deren IP-Adresse (oder Domänenname) einem *Client* entspricht, bestimmt, ob diesem Rechner Zugang gewährt oder verweigert wird. Als Angaben sind „`allow, deny`“, „`deny, allow`“ und „`mutual-failure`“ erlaubt. Im letzten Fall wird der Zugang nur den *Clients* erlaubt, die bei `allow` aber nicht bei `deny`, angeführt sind.

`allow from` erlaubt den angeführten *Clients* Zugang. Die *Clients* können durch ihren (unvollständigen) Domänennamen (z.B. `firma.de`) oder das 1–4 linke Byte der IP-Adressen, z.B. `192.168.33`, angegeben werden.

`deny from` untersagt den angeführten *Clients* Zugang. Die Syntax ist die gleiche wie bei `allow from`.

Beispiel. *Soll der Zugang zu internen Firmendaten, z.B. im* Intranet*, nur lokalen Benutzern gewährt werden, so kann der allgemeine Zugang in der Datei* **access.conf** *auf firmeninterne Nutzer eingeschränkt werden.*

```
1   <Directory /usr/local/docs/firma>
2   <Limit GET POST>
3   order deny,allow
4   deny from all
5   allow from firma.de
6   </Limit>
7   </Directory>
```

[69] die folgenden Zeilennummern beziehen sich auf das Beispiel 2.3.3.3

Der Zugriff aus der genannten Domäne stellt eine Ausnahme des allgemeinen Verbots dar. Wichtig ist dabei die Reihenfolge `deny,allow`*, da im umgekehrten Fall das allgemeine Verbot die letzte Direktive wäre und somit für alle* Clients *gelten würde.*

Da die Direktiven durch `<Limit` methoden`>` *und* `</Limit>` *eingeschlossen werden, beziehen sich die Einschränkungen nur auf die HTTP-Methoden* `GET` *und* `POST`*. Sollen die Einschränkungen für alle Methoden gelten, so ist die* `<Limit>`*-Direktive wegzulassen.* `<Limit>`

Anstelle der oben angegebenen Konstruktion kann auch die nachfolgende verwendet werden.

```
1   <Location /firma>
2   <Limit GET POST>
3   order mutual-failure
4   deny from all
5   allow from 192.168.33
6   </Limit>
7   </Directory>
```

Der Einschluß in `<Location` URL`>` *und* `</Location>` *hat die gleiche Wirkung wie derjenige in* `<Directory` verzeichnis`>` *und* `</Directory`*. Der Unterschied besteht darin, daß sich im ersten Fall die Angabe auf ein konkretes Verzeichnis (den Pfad), im zweiten auf eine URL (den Weg), bezieht.* `<Location>`

In beiden Fällen dürfen zur Namensangabe „wildcards“ und reguläre Ausdrücke, so wie sie z.B. aus UNIX[70] *bekannt sind, verwendet werden.*

Der Zugriff auf HTML-Verzeichnisse kann auch so geregelt werden, daß nur bestimmte Personen ihn nach Eingabe eines Passwortes erhalten. Die Datei mit den Benutzern und ihren Passwörtern sollte außerhalb des Bereichs der HTML-Dokumente liegen, z.B. unter *`ServerRoot`* **/conf**. Sie wird mit Hilfe des Programms **htpasswd** erzeugt.

Beispiel. *Sollen z.B. die Kunden* `Elmar.Ziegler` *und* `Sylvia.Schellberg` *sowie der Angestellte* `Berti.Gaits` *Zugang zu kostenpflichtigen* Server*-Seiten erhalten, so erfolgt ihr Eintrag in die Datei* **.htpasswd** *(der Name ist frei wählbar) wie folgt:*

```
extor:~# htpasswd -c /usr/local/apache/conf/.htpasswd Elmar.Ziegler
extor:~# htpasswd /usr/local/apache/conf/.htpasswd Sylvia.Schellberg
extor:~# htpasswd /usr/local/apache/conf/.htpasswd Berti.Gaits
```

Der Schalter **-c** *legt die Datei an, nach jeder Eingabe fragt das Programm nach dem Passwort, das zuerst chiffriert und dann zusammen mit dem Namen des Benutzers in der Datei abgelegt wird.*

Die Service-Seiten sollen unter *`DocumentRoot`* **/services** *abgelegt werden.*

```
1   <Directory /usr/local/apache/htdocs/services>
2   Options Indexes FollowSymLinks
```

[70] `?` steht für ein beliebiges Zeichen, `*` für eine beliebige Anzahl von Zeichen (0,1,...), reguläre Ausdrücke müssen durch `~` eingeleitet werden

```
3   AllowOverride None
4   AuthName Service
5   AuthUserFile /usr/local/apache/conf/.htpasswd
6   AuthGroupFile /usr/local/apache/conf/.groups
7   require group angestellte kunden
8   </Directory>
```

AuthName

Der Bereich `DocumentRoot` **/services** *soll den Namen* `Service` *erhalten. Dies wird dem* Server *durch die Direktive* `AuthName` (→ Z.4) *mitgeteilt. Der Text „*`Service`*“ wird zusammen mit der Aufforderung, den Namen und das Passwort einzugeben, vom* Browser *ausgegeben.*

AuthUserFile

Die Direktive `AuthUserFile` (→ Z.5) *gibt den absoluten Pfad zur Passwortdatei an.*

AuthGroupF.

Die Benutzer wurden im Beispiel in die Gruppen `angestellte` *und* `kunden` *eingeteilt. Die Gruppen werden in der Datei* **.groups** *definiert. Der Pfad zu ihr wird durch die Direktive* `AuthGroupFile` *festgelegt. Sie enthält folgende Zeilen:*

```
angestellte: Berti.Gaits
kunden: Elmar.Ziegler Sylvia.Schellberg
```

Alternativ zu

`require group angestellte kunden` (→ Z.7)

könnte man auch

```
require user Berti.Gaits Elmar.Ziegler Sylvia.Schellberg
```

schreiben, wobei dann die Direktive `AuthGroupFile` *entfallen kann.*

2.3.3.4 Virtueller Server

Apache unterstützt auch die Bildung sogenannter virtueller *Server*. Auf einem Rechner können so mehrere unabhängige WWW-*Server* eingerichtet werden. Dies kann dadurch geschehen, daß für jeden virtuellen *Server* ein eigener *Daemon* gestartet wird oder ein *Daemon* mehrere virtuelle *Server* realisiert.

Die einzelnen *Server* können entweder über unterschiedliche IP-Adressen oder Aliasnamen (→ S.33) angesprochen werden.

Hat der Rechner mehrere Netzkarten, so können die entsprechenden IP-Adressen verwendet werden. Ist nur eine Netzkarte vorhanden, so können dieser Netzschnittstelle weitere IP-Adressen, sogenannte Alias-IP, vergeben werden.

Beispiel. *Angenommen, der Rechner* `antje` *sei der primäre WWW-*Server*. Jetzt soll auf ihm zusätzlich ein virtueller* Server *eingerichtet werden. Dazu muß zuerst neben* `eth0` *eine virtuelle Netzschnittstelle* `eth0:0` *eingerichtet und dieser ein IP-Alias zugeordnet werden.*

```
antje:~# ifconfig eth0:0 192.168.33.130 netmask 255.255.255.224 broadcast 192.168.33.95
antje:~# route add -host 192.168.33.130 dev eth0:0
```

Nachdem der IP-Alias `192.168.33.130` *der Netzschnittstelle* `eth0:0` *zugeordnet wurde, muß die Adresse noch in die Zonendatei der betreffenden Zone eingetragen werden* (→ S.31). *Soll der virtuelle* Server *z.B.* `info.firma.de` *heißen, so lautet der Eintrag*

```
info        IN      A       192.168.33.130
```

Nun müssen noch in der Datei **httpd.conf** *folgende Zeilen hinzugefügt werden:*

```
1   <VirtualHost 192.168.33.130>
2   ServerAdmin webmaster@info.firma.de
3   DocumentRoot /usr/local/apache/htdocs/info
4   ServerName info.firma.de
5   ErrorLog logs/info.firma.de-error_log
6   TransferLog logs/info.firma.de-access_log
7   </VirtualHost>
```

Die Direktiven `<VirtualHost` [IP-Adresse|Domänenname]`>` *und* `</VirtualHost>` *schließen Direktiven ein, die für den entweder durch die IP-Adresse oder den vollqualifizierten Domänennamen spezifizierten virtuellen* Server *gelten sollen. Insbesondere ist dies die Direktive* `DocumentRoot`*, durch welche für die unterschiedlichen* Server *jeweils andere Wurzeln für die Verzeichnisse, in denen die HTML-Seiten abgelegt sind, eingestellt werden können.* <VirtualH.>

Soll der virtuelle Server *auf dem Rechner* `extor` *eingerichtet werden, so kann dazu seine zweite Netzschnittstelle verwendet werden. In die Zonendatei der Zone* `firma.de` (→ S.31) *ist dann statt der obigen folgende Zeile einzutragen:*

```
info        IN      A       172.24.233.253
```

Die Konfigurationsdatei **httpd.conf** *ist nun wie folgt zu erweitern:*

```
1   <VirtualHost 192.168.33.94>
2   ServerAdmin webmaster@www.firma.de
3   DocumentRoot /usr/local/apache/htdocs
4   ServerName www.firma.de
5   ErrorLog logs/error_log
6   TransferLog logs/access_log
7   </VirtualHost>
8
9   <VirtualHost 172.24.233.253>
10  ServerAdmin webmaster@info.firma.de
11  DocumentRoot /usr/local/apache/htdocs/info
12  ServerName info.firma.de
13  ErrorLog logs/info.firma.de-error_log
14  TransferLog logs/info.firma.de-access_log
15  </VirtualHost>
```

Um DNS-Anfragen zu vermeiden, sollten aus den Angaben immer sowohl die IP-Adresse als auch der Domänenname ersichtlich sein.

Erfolgt der Zugang zu den virtuellen *Servern* über die Aliasnamen, so kann dies ebenfalls über eine oder mehrere IP-Adressen geschehen. Beide Vorgehensweisen können auch kombiniert werden.

Ein virtueller *Server* kann auch dadurch realisiert werden, daß man einer IP-Adresse unterschiedliche *Ports* zuordnet (→ S.137).

2.3.3.5 Stellvertretender WWW-Server

Firewall

Im nächsten Kapitel werden Möglichkeiten beschrieben, wie ein Netz mit Hilfe einer *Firewall* gesichert werden kann. Solch eine *Firewall* schränkt das *Routing* ein. Die einzigen, die Zugang zum externen (*Internet*) und lokalen Netz haben, sind Programme, die auf dem Rechner laufen, auf dem die *Firewall* realisiert ist. Dies sichert zwar das lokale Netz gegen Eindringlinge von außen, verhindert aber auch den Zugang lokaler Benutzer zum externen Netz. Diese können sich mit dem externen Netz nur über stellvertretende Applikationen, die auf dem *Firewall*-Rechner laufen und *Proxy Applications* bzw. *Proxy Server* genannt werden, in Verbindung setzen. Apache kann die Funktion[71] solch einer *Proxy Application* für WWW und FTP übernehmen[72].

Die *Proxy Application* von Apache hat neben der Vermittlungstätigkeit eine weitere nützliche Funktion — die Zwischenspeicherung der vermittelten Dokumente in einem lokalen Speicher (*Cache*). Nachfolgende Anfragen lokaler Benutzer nach dem gleichen Dokument können so in der Regel direkt aus diesem Speicher befriedigt werden, eine erneute Übertragung des Dokuments vom entfernten WWW-*Server* ist nicht notwendig.

Eine erneute Übertragung[73] erfolgt nur dann, wenn der WWW-*Server* durch Vergleich der Angabe `If-Modified-Since` im Kopf der Anfrage und dem Zeitpunkt der letzten Änderung des Dokuments feststellt, daß sein Dokument aktueller ist als dasjenige im *Cache* des *Proxy Servers*. Ist dies nicht der Fall, so schickt er dem *Proxy Server* im Kopf der Antwort lediglich die Statuszeile `304` (`not modified`) und nicht das Dokument.

Die Konfiguration von Apache als *Proxy Server* erfolgt mit Hilfe von Direktiven, die in der Datei **httpd.conf-dist** mit einem Kommentarzeichen versehen, bereits vordefiniert sind.

Beispiel.

```
1  ProxyRequests On
2  CacheRoot /usr/local/apache/proxy
3  CacheSize 5
4  CacheGcInterval 4
5  CacheMaxExpire 24
6  CacheLastModifiedFactor 0.1
7  CacheDefaultExpire 1
8  NoCache xxxservers.com
```

ProxyR.

Durch die Direktive `ProxyRequests On` *wird die Funktion* Proxy Application *einge-*

[71] nur nach vorherigem Auskommentieren in `src/Configuration` und erneutem Übersetzen

[72] weiterreichende Möglichkeiten hat z.B. das Programm Squid (`http://squid.nlanr.net/Squid/`), das ebenfalls als *Proxy Application* für WWW, FTP und Gopher eingesetzt werden kann

[73] mit einem aktuellen Wert für `Last-Modified` in der Kopfzeile der Antwort

schaltet. `CacheRoot` *gibt den Pfad zum Verzeichnis an, in dem der Dokumentespeicher der Größe* `CacheSize` *(in Kilobyte) angelegt wird. Wird die angegebene Länge der Dokumente überschritten, so sorgt eine Prozedur dafür, daß so viele Dokumente gelöscht werden, bis die erlaubte Größe wieder erreicht ist. Der Gesamtumfang des* Cache *wird dazu alle* `CacheGcInterval` *Stunden kontrolliert.* `CacheRoot` `CacheSize` `CacheGcI.`

Unabhängig von eventuellen Angaben im Dokument wird es nach `CacheMaxExpire` *Tagen aus den Speicher wieder gelöscht. Enthält das Dokument kein Verfallsdatum, so wird es geschätzt. Dies geschieht mit Hilfe der Direktive* `CacheLastModifiedFactor`*. Ergibt die Schätzung einen höheren Wert als* `CacheMaxExpire`*, so wird der letztere verwendet. Wird das Dokument von einem HTTP-Protokoll geholt, das keine Verfallszeiten auswerten kann, so wird* `CacheDefaultExpire` *verwendet. Durch* `NoCache` *wird bestimmt, welche Seiten nicht gespeichert werden sollen.* `CacheMaxE.` `CacheLMF.` `CacheDE.` `NoCache`

2.3.3.6 WWW-Zugriff via HTTPS

Ein *Secure*-HTTP-*Server* benutzt die selben Dokumente und die selbe Konfiguration wie der normale Apache-*Server*. Der Unterschied besteht darin, daß ein *Secure*-HTTP-*Server*, wie z.B. Stronghold[74], zur Verschlüsselung zwischen HTTP und TCP das Protokoll SSL (*Secure Sockets Layer*) einfügt. Dieses Protokoll dient zur Verschlüsselung sämtlicher Informationen, die während einer Verbindung übertragen werden, also sowohl der HTTP-Anfrage und -Antwort als auch der URL, sowie der in Formularen übermittelten Informationen, wie z.B. der Nummer einer Kreditkarte, des Benutzernamens und Passworts. Vor dem Verbindungsaufbau zwischen *Client* und *Server* erfolgt eine Sicherheitsüberprüfung wobei eine Sicherheitsstufe für die Übertragung festgelegt wird. SSL

Stronghold unterstützt z.B. die Versionen 2 und 3 von SSL und verwendet eine Schlüssellänge von 128 Bit.

Um SSL im *Internet* verwenden zu können, braucht der Betreiber des *Servers* ein Zertifikat einer Zertifizierungsstelle.

Dokumente, die über *Secure*-HTTP angefordert werden sollen, müssen mit dem Schema `https` adressiert werden.

Beispiel.
`https://www.lufthansa.com/on-line-schedule/?profile=0&startFrame=pm_2010010`

Soll der Zugang über normales HTTP unterbunden werden, so muß in der Datei **.htaccess** im Verzeichnis, in dem das Dokument abgelegt ist, die Direktive `RequireSSL on` eingetragen werden. `RequireSSL`

2.4 USENET

Die Verteilung von Bulletins, Informationen und Daten erfolgte bereits an die Benutzer des ARPANET auf eine schnelle und effiziente Art und Weise.

[74] Stronghold (`http://www.int.c2.net/stronghold/`) ist ein *Secure-Server*, der auf Apache basiert

Heute werden im *Internet* vorwiegend zwei Methoden zur Verteilung solcher Art von Nachrichten benutzt — *Mailing*-Listen und USENET-*News*[75].

In elektronischen Konferenzen (→ S.55) wird jeder Beitrag in eine Konferenz an alle anderen Mitglieder verteilt. Falls die Anzahl der Teilnehmer stark zunimmt, steigt dadurch auch der Bedarf an notwendiger Bandbreite und CPU-Zeit, so daß das Verfahren sehr schnell ineffizient wird. Gleichzeitig steigt der Bedarf des insgesamt benötigten Speicherplatzes, da jeder Beitrag auf jedem Teilnehmerrechner abgelegt werden muß.

Probleme können auch bei der Pflege der Listen auftreten. Teilnehmer können z.B. ihre Adresse ändern, sie können sich abmelden, neue können hinzukommen, *Clients* können ihren Betrieb einstellen bzw. neue können den Betrieb aufnehmen.

Eine Lösung für die genannten Probleme bietet USENET. Hier werden Beiträge zentral auf einem *Server* gehalten und der Benutzer kann sie sich nach Bedarf auf seinen Rechner herunterladen. Ebenso hat er die Möglichkeit, an diesen *Server* eigene Beiträge zu schicken, die dann von diesem an weitere *News-Server* automatisch weitergeleitet werden.

Zur Übertragung der Beiträge stehen zwei Protokolle zur Verfügung, das ältere UUCP (→ Sn.42,52), das bereits im ARPANET Verwendung fand und NNTP (→ S.74).

UUCP UUCP wird vorwiegend zur Fernübertragung von Daten verwendet. Es ist das einzige Protokoll, das zu diesem Zweck eine Verbindung über Modem gestattet, es funktioniert aber auch über eine feste TCP-Verbindung. Der Vorteil von UUCP liegt darin, daß der Austausch von Daten auf bestimmte Zeiten beschränkt werden kann (z.B. mit Hilfe von **cron**). Zur Übertragung von Daten (z.B. Post, *News*) stellt der lokale *Server* eine UCCP-Verbindung mit einem *Provider* her und tauscht alle lokalen Daten mit den beim *Provider* angesammelten aus, wonach die Verbindung wieder unterbrochen wird.

Aus Sicherheitsgründen (→ S.52) sollte UUCP für Verbindungen aus dem internen Firmennetz in das *Internet* nicht mehr verwendet werden.

NNTP Im allgemeinen wird heute zur Übertragung von USENET-*News* im *Internet* NNTP verwendet. Der *News-Server* ist dabei die Schnittstelle, über die der Nachrichtenaustausch erfolgt. Es stehen unterschiedliche *News-Server*[76] zur Verfügung, die untereinander über NNTP kommunizieren. Dabei wird besonderer Wert darauf gelegt, daß der *Server* nicht allzu weit vom *Client* entfernt liegt, falls möglich also beide über das lokale Netz miteinander verbunden sind. Gründe dafür können die Geschwindigkeit beim Datenzugriff aber auch die Sicherheit sein. Oft is es daher so, daß die Systemadministratoren den *News-Server* so konfigurieren, daß er nur den Empfang von Beiträgen aus dem Firmennetz gestattet, wogegen die angebotenen Nachrichten von allen gelesen werden können. Aber auch dieses Angebot kann aus rein politischen oder aus Kapazitätsgründen eingeschränkt werden.

News werden in eine Handvoll übergeordneter Hierarchien, sogenannte Diskussionsforen (*Newsgroups*), eingeteilt.

[75] auch *Network News* oder *News* genannt
[76] z.B. *B News*, *C News* und INN (*InterNetNews*)

Gruppe	Thema
`alt`	Verschiedenes
`bit`	Kopien von elektronischen Konferenzen aus dem BITNET
`biz`	Wirtschaft
`comp`	Rechner
`humanities`	Humanwissenschaften
`misc`	alles, was sich nicht anders einordnen läßt
`news`	über *News*
`rec`	Erholung
`sci`	Wissenschaft
`soc`	Sozialwissenschaft
`talk`	heiße Themen
`de`	deutschsprachige Gruppen

Tab. 2.23: *Newsgroups* der höchsten Hierarchiestufe

Jedem Diskussionsforum wird ein bestimmtes Thema (→ Tab.2.23) zugeordnet. Die Spezifizierung des Themas erfolgt durch das Anhängen von Untergruppen, z.B. `comp`, `comp.os`, `comp.os.linux`, usw.

Für die Einrichtung neuer Diskussionsforen im offiziellen Teil des USENET gibt es ein festes Reglement. Über jeden Vorschlag zur Neueinrichtung einer Gruppe oder Untergruppe wird diskutiert und abgestimmt. Dies gilt nicht für die Gruppe `alt`. In diesem Diskussionsforum können Gruppen von den einzelnen Benutzern eigenmächtig ins Leben gerufen werden.

Die Gruppe `alt` wurde von Brian Reid aus Protest eingeführt, nachdem der damalige *News*-Guru Gene Spafford sich aus Anstandsgründen geweigert hat, in den offiziellen Hierarchien die Untergruppen `soc.sex` und `soc.drugs` einzurichten. Reid hat deshalb 1988 die Gruppe `alt` und gleichzeitig die Untergruppen `alt.sex`, `alt.drugs` und `alt.rock-n-roll`, eingerichtet.

1997 gab es weltweit mehr als 15000 *Newsgroups*. Da dazu auch Gruppen in den jeweiligen Landessprachen zählen (`de` — deutsch, `cz` — tschechisch, usw.), haben manche davon sicher nur geographisch begrenzte Bedeutung und müssen nicht allen Benutzern zur Verfügung gestellt werden. Das tägliche Datenaufkommen lag 1997 bei über einem GByte, wovon 85% auf `alt`, 6% auf `rec`, 3% auf `comp` und 2% auf `soc`, entfielen.

Möchte man alle Diskussionsforen abonnieren, so liegt das Hauptproblem sicher in der Menge der täglich anfallenden Daten. Dies macht den Betrieb von USENET-*News* zu einer sehr aufwendigen und anspruchsvollen Aufgabe.

2.4.1 InterNetNews Server

Der *News-Server* INN (*InterNetNews*[77]) besteht aus einer Familie gegenseitig zusammenarbeitender Programme, deren Koordination durch den eigentlichen *News-Server*, der durch das Programm **innd** realisiert wird, erfolgt. **innd**

[77] `http://www.cis.ohio-state.edu/~barr/INN.html`,
Dokumentation unter `http://www.mibsoftware.com/userkt/userkt.html`

Dieses Programm überwacht den NNTP-*Port* 119. Falls an diesem *Port* ein externer Rechner eine Verbindung anknüpft, so überprüft es seine Identität anhand der Einträge in der Datei **hosts.nntp**. Hat der Rechner dort einen Eintrag, so handelt es sich bei ihm um einen *News-Server* (nur für diese ist **innd** zuständig) und **innd** übernimmt von ihm neue Beiträge in die *Newsgroups*. Besteht kein Eintrag in **hosts.nntp**, so geht **innd** davon aus, daß es sich bei dem Rechner um einen *Client* handelt und überläßt die Kommunikation dem für *Clients* zuständigen *Server* **nnrpd**.

nnrpd

Sobald **innd** eine neue Nachricht erhalten hat (gleichgültig, ob von einem anderen *Server* oder einem *Client*), so versucht er sie weiterzugeben. Diesem Zweck dienen die Datei **newsfeeds**, in der die Wege zu benachbarten *News-Servern* eingetragen sind und das Programm **nntpsend**, das für das Abschicken der Nachrichten verantwortlich ist. **nntpsend** wird dazu in regelmäßigen Intervallen vom *Daemon* **cron** aufgerufen, die Nachricht, die verschickt werden soll, befindet sich im Unterverzeichnis **/var/spool/news/out.going**.

nntpsend

Alle bisher angeführten Programme und Konfigurationsdateien befinden sich im Unterverzeichnis **/usr/lib/news**.

Zum Betrieb des *News-Servers* sollte ein eigener Benutzer **news** und eine gleichnamige Gruppe eingerichtet werden. Da bei manchen Programmen zur Administration die Meldungen per *e-mail* an den Benutzer **usenet** erfolgen, sollte dem Benutzer **news** zusätzlich der Aliasname **usenet** zugeordnet werden.

Der *Server* **innd** wird beim Hochfahren des Systems mit Hilfe des Skripts **rc.news**, das von **/etc/rc.d** aus gestartet wird, in Betrieb genommen. Für den ordnungsgemäßen Betrieb wird dann eine Datei **crontab** benötigt, die für den Benutzer **news** mit nachfolgendem Inhalt angelegt werden muß.

rc.news

```
0,15,30,45 * * * * /usr/lib/news/bin/nntpsend
0 22 * * * /usr/lib/news/bin/news.daily < /dev/null
```

Der erste Eintrag sorgt viertelstündlich dafür, daß neue Nachrichten verschickt werden, der zweite dient der täglichen Wartung. Diese beginnt um 22^{00} h und ihre Hauptaufgabe besteht darin, die Archiveinträge zu aktualisieren. Informationen über angekommene Beiträge werden in der Datei **log**, über Fehler in der Datei **errlog**, protokolliert. Beide Dateien befinden sich im Unterverzeichnis **/usr/lib/news**.

Beispiel. *In den nachfolgenden Beispielen wird davon ausgegangen, daß die Firma von Seite 11 einen* News-Server *auf dem Rechner* `news.firma.de` *betreibt. Die Verbindung zum Rest der Welt erfolgt über* `news.extern.de`*. Auf dem Firmenserver existieren neben den üblichen Diskussionsforen auch rein lokale Gruppen, die nicht über den* Server `news.extern.de` *verteilt werden sollen.*

Diese Gruppen könnten z.B. `firma.projekte` *oder* `firma.aktuell.mitarbeiter`*, heißen.*

Neben `news.firma.de` *soll es noch einen mit einer Standleitung verbundenen Hilfsserver namens* `mini.firma.de` *geben, der nur die Gruppen* `firma.*` *und* `comp.os.linux.*` *bereitstellen soll (→ Abb. 2.19).*

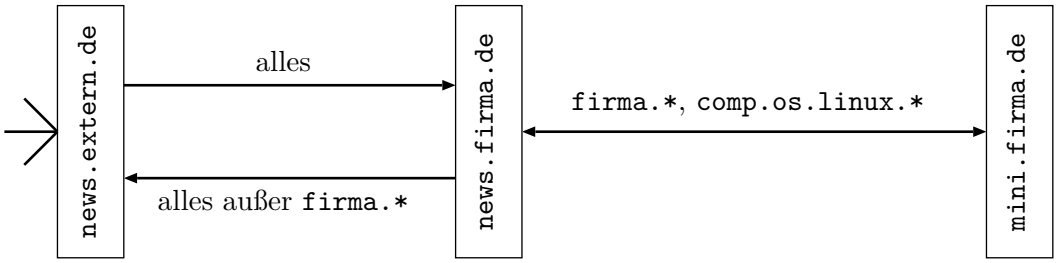


Abb. 2.19: Fluß der Nachrichten in der Firma

2.4.1.1 Kommunikation mit anderen News-Servern

Wie bereits weiter oben gesagt, übernimmt die Kommunikation mit anderen *News-Servern* **innd**. Seine Konfigurationsdatei **inn.conf** enthält Einträge, die den Nachrichten, die auf diesem *Server* entstehen, als Kopfzeilen hinzugefügt werden. **inn.conf**

Beispiel.

```
server:         news.firma.de
domain:         firma.de
organization:   Firma GmbH
```

Die meisten der ca. 10 möglichen Einträge sind jedoch überflüssig, da **innd** selbst in der Lage ist, sie aus der Systemkonfiguration zu ermitteln. Eine Ausnahme bildet die Zeile `organization`. Sie enthält den Namen der Quelle, wo die Nachricht entstanden ist und sollte daher nie fehlen.

Die Datei **hosts.nntp** enthält die Namen der *News-Server*, von denen **innd** bereit ist, *News* zu empfangen. Jede Zeile entspricht einem Rechner und hat folgenden Form: **hosts.nntp**

adresse:*passwort*:*gruppen*

Die *adresse* kann der Rechnername oder seine IP-Adresse sein, das *passwort* und die *gruppen* sind optional. Die Angabe *gruppen* bestimmt, welche Gruppen von dem gegebenen Rechner akzeptiert werden sollen.

Beispiel.

```
news.extern.de:
mini.firma.de:
```

An welche *News-Server* Nachrichten verschickt werden, steht in der Datei **newsfeeds**. Die Zeilen haben folgendes Aussehen: **newsfeeds**

name[/*aliasnamen*]\
:*gruppen*[/*distributionen*]\
:[*schalter*]\
:[*parameter*]

Der umgekehrte Schrägstrich gefolgt vom Ende der Zeile wird als Leerzeichen interpretiert (alle Angaben stehen folglich in einer einzigen Zeile).

Der *name* definiert den Namen des Weges zum benachbarten *News-Server*, die einzelnen *aliasnamen* geben die unterschiedlichen Namen an, unter denen der Rechner in der

Kopfzeile `Path` erscheinen kann. Findet **innd** *name* oder *aliasname* im `Path`, so verschickt er die Nachricht an diesen Rechner *nicht*, da sie ja bereits über ihn gegangen ist.

In der zweiten Zeile stehen die Namen der Gruppen, die über diesen Weg geschickt werden sollen. Beliebige Zeichenketten werden dabei durch `*` angegeben, ein führendes Ausrufezeichen gibt an, daß die entsprechende Gruppe nicht verschickt werden soll.

Beispiel. *Auf dem* Server `news.firma.de` *kann die Datei* **newsfeeds** *z.B. folgendes Aussehen haben:*

```
extern/news.extern.de\
        :*,!firma.*/!firma\
        ::

mini/mini.firma.de\
        :firma.*,comp.os.linux.*\
        ::
```

Auf dem Server `mini.firma.de` *muß nur der Weg zum* News-Server `news.firma.de` *definiert werden. Auf diesem Weg werden alle Beiträge verschickt.*

```
firma/news.firma.de\
        :*/*\
        ::
```

moderators

Eine wichtige Rolle beim Verschicken von Nachrichten spielt die Datei **moderators**. Sie enthält die *e-mail*-Adressen der Verwalter moderierter Konferenzen.

Bei der Entstehung einer neuen Konferenz wird angegeben, ob sie moderiert werden soll oder nicht. Dabei wird jedoch die *e-mail*-Adresse des Moderators nicht festgelegt, sie wird erst bei Bedarf aus der Datei **moderators** ermittelt. Die Zeilen der Datei **moderators** haben folgende Form:

gruppe: *e-mail-adresse-des-moderators*

Die Moderatoren einer Hierarchie von Diskussionsgruppen haben eine gemeinsame Domäne, ihre Namen werden aus den Namen der Untergruppen abgeleitet.

Beispiel. *Für moderierte Gruppen von GNU gilt z.B.*

```
gnu.*:%s@tut.cis.ohio-state.edu
```

In der Adresse wird `%s` *durch den Namen der Gruppe ersetzt, deren Moderator gesucht wird. Dabei werden lediglich die Punkte durch Bindestriche ersetzt, also z.B. für die Gruppe* `gnu.emacs` *die Adresse* `gnu-emacs@tut.cis.ohio-state.edu` *gebildet.*

Beiträge in moderierte Gruppen gehen folglich nicht direkt in das Diskussionsforum sondern werden per *e-mail* an den Moderator zur Begutachtung geschickt. Falls dieser den Beitrag für akzeptabel befindet, so fügt er die Kopfzeile `Approved` hinzu und schickt ihn in die entsprechende Gruppe auf seinem lokalen *News-Server*. Dieser verteilt sie dann auf dem üblichen Weg weiter.

innxmit

Das eigentliche Verschicken der *News* besorgt **nntpsend**. Er tut dies allerdings nicht selbst, sondern übergibt den Auftrag an das Programm **innxmit**.

Die wichtigste Konfigurationsdatei von **nntpsend** ist **nntpsend.ctl**. In ihr werden den *namen* der Wege zu benachbarten *News-Servern*, so wie sie in der Datei **newsfeeds** stehen, die konkreten Domänennamen der Rechner zugeordnet. **nntpsend.ctl**

name: *domänenname-des-rechners*: *größe*: *parameter*

Beispiel.
```
extern:news.extern.de::
mini:mini.firma.de::
```

Wird vom entfernten *Server* eine Identifizierung verlangt (→ *passwort* in **hosts.nntp**), so ist diese in der Form

rechnername: *benutzername*: *passwort*: *stil*

in die Datei **passwd.nntp** einzutragen. Der *stil* legt die Art der Identifizierung fest, implizit wird `authinfo` angegeben[78]. **passwd.nntp**

2.4.1.2 Kommunikation mit Clients

Den Kontakt zu den *Clients* besorgt **nnrpd**. Die Zugriffsrechte werden in der Datei **nnrp.access** geregelt. Eine Zeile dieser Datei hat folgendes Aussehen: **nnrp.access**

rechnername: *tätigkeit*: *benutzer*: *passwort*: *gruppen*

Der *rechnername* ist eine Schablone, die auf die Domänennamen der *Client*-Rechner gelegt wird. Die letzte Zeile, die der Schablone genügt, bestimmt die Zugriffsrechte. Die Zeilen sollten daher von allgemeineren zu konkreten angeordnet werden.

Für *tätigkeit* können zwei Worte angegeben werden — `Read` und `Post`. Sie bestimmen, ob der *Client* Nachrichten lesen und schicken darf.

Die Identifikation des Autors erfolgt über *benutzer* und *passwort*. Die beiden Einträge können zur Zugangsbeschränkung verwendet werden. Steht z.B. anstelle von *passwort* ein Leerzeichen, so hat der *Client* keine Möglichkeit, solch ein Passwort einzugeben und wird daher nicht zum *Server* zugelassen.

Der letzte Eintrag enthält die *gruppen*, die diese Einschränkungen betreffen.

Beispiel. *Eine konservative Strategie — der lokale Benutzer darf lesen und Beiträge schicken, alle anderen dürfen nichts — hätte folgende* **nnrp.access** *zur Folge:*

```
*:: : :!*
*.firma.de:Read Post:::*
```

Eine etwas liberalere Version ist folgende:

```
*:Read:::*,!firma.*
*.kumpels.de:Read Post:::firma.*
*.firma.de:Read Post:::*
```

Die lokalen Benutzer dürfen wieder alles (letzte Zeile), die Benutzer einer befreundeten Firma dürfen mit den Firmengruppen arbeiten und der Rest der Welt darf mit Ausnahme der Firmengruppen alle anderen lesen.

[78] RFC 977

Schalter	Bedeutung
`y`	lokale Benutzer dürfen Beiträge in die Gruppe schicken
`n`	nur externe Benutzer dürfen Beiträge in die Gruppe schicken
`m`	die Gruppe wird moderiert
`j`	Beiträge in diese Gruppe werden nicht gespeichert, sie werden an *Server* weitergereicht, die diese Gruppe abonniert haben; lokale Beiträge in diese Gruppe sind verboten
`x`	in diese Gruppe können keine Beiträge geschickt werden
`=`*reelle_gruppe*	*gruppe* ist ein Aliasname für *reelle_gruppe*, Beiträge in *gruppe* gehen an *reelle_gruppe*

Tab. 2.24: Schalter, die das Gruppenverhalten beeinflussen

An dem Beispiel sieht man recht deutlich, welch eingeschränkte Möglichkeiten man hat, die Zugriffsrechte festzulegen. Da für jeden Client *immer nur eine Zeile maßgebend ist, ist z.B. für die befreundete Firma nicht gleichzeitig möglich, bestimmte Gruppen nur zum Lesen freizugeben und in andere zu erlauben, Beiträge zu schicken.*

2.4.1.3 Instandhaltung des Servers

active In der Datei **/news/lib/active** stehen alle Gruppen, die auf dem *Server* vertreten sein sollen. Sie werden zeilenweise angegeben, jede Zeile enthält folgende Angaben:

gruppe max.-beitragsnummer min.-beitragsnummer schalter

Die *Newsgroup* wird durch *gruppe* festgelegt. Die beiden Zahlen geben an, in welchem Intervall sich die durchnumerierten Beiträge der angegebenen *gruppe* bewegen dürfen. Die *schalter* beeinflussen das Verhalten der Gruppe (→ Tab.2.24).

Gruppen, die nicht in **active** aufgeführt sind, existieren für den *Server* nicht. Um die Datei zu erzeugen wird empfohlen, sich mit dem *Server*, von dem die *News* abgenommen werden sollen, zu verbinden und zunächst seine Datei zu kopieren.

```
extor:~# telnet news.extern.de 119 >active.tmp
```

Durch die Eingabe von `list` wird die Datei **active.tmp** erzeugt. Diese kann editiert, den eigenen Bedürfnissen angepaßt und anschließend nach **/news/lib/active** kopiert werden. Die Datei **active** kann während dem Betrieb durch die *Server*, die Nachrichten schicken, modifiziert werden. Manuelle Eingriffe erfolgen mit Hilfe des Programms **ctlinnd**. Als Parameter werdem ihm z.B. folgende Kommandos und Argumente übergeben:

ctlinnd

`changegroup` *gruppe schalter*
`newgroup` *gruppe schalter gründer*
`rmgroup` *gruppe*

Das erste Kommando ändert bei der *gruppe* in der Datei **active** die aktuellen *schalter*. Das zweite erzeugt eine neue *gruppe* und trägt sie in **active** ein. Das dritte macht das Gegenteil.

Beispiel. *Durch*

```
extor:~# ctlinnd newgroup firma.info y "Christian Walch"
extor:~# ctlinnd newgroup firma.katalog ym "Christian Walch"
```

werden zwei neue lokale Gruppen eingerichtet, wobei die zweite moderiert wird. Diese Gruppe muß zusätzlich in die Datei **moderators** *eingetragen werden.*

Zum Entfernen eines Beitrags gibt es für **ctlinnd** das Kommando **cancel**, dem als Argument der Identifikator der Nachricht übergeben wird.

Eine zentrale Rolle bei der täglichen Instandhaltung spielt das Programm **news.daily**. Seine Hauptaufgabe besteht darin, abgelaufene Nachrichten zu löschen. **news.daily**

Für jeden Beitrag einer Gruppe ist eine implizite Lebensdauer definiert. Die Lebensdauer wird in der Datei **expire.ctl** festgelegt. Ihre Zeilen sehen wie folgt aus: **expire.ctl**

gruppen:*typ*:*minimum*:*implizit*:*maximum*

gruppen gibt wiederum an, auf welche Gruppen sich die nachfolgenden Einschränkungen beziehen. Der *typ* bestimmt, ob sie nur für moderierte (`M`), unmoderierte (`U`) oder aber für alle Gruppen (`A`) gelten sollen. Die restlichen Angaben legen die Lebensdauer der *gruppen* fest. Falls nicht explizit durch die Angabe `Expires` im Kopf der Nachricht anders angegeben, wird die Nachricht eine *implizite* Anzahl von Tagen behalten. Ist `Expires` vorhanden, so versucht der *Server*, die sich daraus ergebende Zeit einzuhalten. Er löscht sie jedoch nicht vor *minimum* und nicht später als nach *maximum* Tagen, auch wenn `Expires` eine kürzere bzw. längere Zeitspanne angibt. Wird statt der Anzahl von Tagen `never` angegeben, so behält der *Server* die Nachricht eine unbefristete Zeit.

Durch

`/remember/:`*tage*

wird dem *Server* gesagt, wie lange er aus den *Newsgroups* gelöschte Beiträge noch aufbewahren soll. Dadurch wird vermieden, daß bereits gelöschte *News*, die dem *Server* aus irgendeinem Grund in kurzem Abstand nochmal zugestellt werden, erneut den Benutzern als neue Nachrichten vorgelegt werden.

Beispiel. *Die allgemeine Voreinstellung für die Lebensdauer der Gruppen soll 7 Tage mit einer Abweichung von 1-14 Tagen sein. In den Gruppen* `alt.binaries`*, die wegen der oft vorhandenen (obszönen) Bilder sehr umfangreich geraten können, wird der Wert auf 2 Tage mit einer Abweichung von 1-5 Tagen verkürzt, bei moderierten Gruppen wird der implizite Wert genommen. Firmengruppen werden bevorzugt und daher auf einen hohen Wert gesetzt.*

```
/remember/:14
*:A:1:7:14
alt.binaries.*:U:1:2:5
alt.binaries.*:M:1:7:14
*firma.*:A:1:30:never
```

In der Datei **/news/lib/history** werden die Beiträge, die über den *Server* in letzter Zeit gegangen sind, protokolliert. Die Aktualisierung der Datei erfolgt durch das Programm **expire**, das im Rahmen der täglichen Instandhaltung vom Programm **news.daily** auf- **history** **expire**

gerufen wird.

Ein Restart des *News-Servers* erfolgt durch

```
extor:~# ctlinnd reload all ""
```

3 Sicherheit in TCP/IP-Netzen

Dieses Kapitel behandelt einen Teilaspekt einer umfangreichen Sicherheitspolitik, die sich gegen sozial und mental zurückgebliebene Individuen, sogenannte *Hacker*, richtet. Diese dringen in fremde Systeme nicht nur zu ihrem zweifelhaften Vergnügen, sondern auch mit konkreten Zielen sich zu bereichern, zu rächen oder zu spionieren, ein.

Hacker

Die TCP/IP Protokollfamilie enthält keine Möglichkeiten, sich gegen solche Attacken zu schützen. Sie wurde dazu entwickelt, dem Benutzer den Zugang zu entfernten Rechnern so einfach wie möglich zu machen, auch auf Kosten der Sicherheit.

Jede einzelne Station im Netz kann mit Hilfe des TCP-*Daemons* **tcpd** gesichert werden (→ S.61).

In größeren Netzen wäre es jedoch viel zu kompliziert, den Zugang zu den Netzdiensten auf jeder einzelnen Station zu regeln. Viel sinnvoller ist es, die Zugriffsrechte für ein komplettes Netz zentral einzustellen. Rechner, die zu diesem Zweck verwendet werden, nennt man *Firewalls*. Eine *Firewall* teilt das *Internet* in ein inneres geschütztes und ein äußeres öffentliches Netz auf. Die Kontrolle des Datenstroms zwischen diesen Netzen kann auf unterschiedliche Art und Weise geschehen.

Firewall

Die wichtigsten Kontrollmechanismen sind folgende:

Packet Filter sind normalerweise *Router*, die Pakete nach bestimmten Regeln weiterleiten bzw. herausfiltern. Diese Regeln können z.B. auf die IP-Adressen (sowohl auf die des Senders als auch auf die des Adressaten) oder die Dienste (genauer ihre *Ports*) angewendet werden.

Proxy Server leiten überhaupt keine Pakete weiter. Sie bilden eine Bastion (daher auch der Name *Bastion Host*) gegen die „feindliche“ Welt außen. Möchte ein *Client* aus dem geschützten Netz mit einem *Server* im *Internet* kommunizieren, so richtet er seine Anfrage an den *Proxy Server*, der auf dem *Firewall*-Rechner läuft. Ist der *Client* berechtigt, diesen Dienst anzufordern, überreicht der *Proxy Server* seine Anfrage an den Zielserver und leitet die Antwort an den *Client* weiter. Da die Aktion auf Applikationsebene abläuft, kann sie in allen Schritten protokolliert werden.

Bastion Host

Im folgenden werden die beiden Konzepte kurz besprochen. Als Referenznetz dient wiederum das Firmennetz von Seite 11. Wie bereits in der Fußnote auf Seite 8 erwähnt, werden in dem internen Netz private, nicht im *Internet* zugelassene Adressen verwendet. Damit die Rechner aus dem Firmennetz dennoch mit Stationen im *Internet* kommunizieren können, müssen ihre privaten Adressen nach außen als gültige *Internet*-Adresse erscheinen. Dafür sorgt in unserem Fall der Rechner `verbinde.extern.de`. Er ist als *Packet Filter* mit der Eigenschaft IP-*Masquerading*[1] eingerichtet.

Masquerading

[1] auch als *address hiding* bezeichnet

Bei allen Datagrammen, auf die IP-*Masquerading* angewendet wird, erfolgen folgende Korrekturen:

- Die private IP-Adresse des Senders wird durch die gültige *Internet*-Adresse des Rechners ersetzt, auf dem der *Packet Filter* läuft.
- Der *Port* des Senders wird durch einen ephemeren *Port*[2] ersetzt.

Der *Packet Filter* merkt sich bei allen so generierten ephemeren *Ports* die zugehörige IP-Adresse und den *Port* des ursprünglichen Datagramms, so daß er Datagramme, die er von der Empfängerseite aus dem *Internet* zurückerhält, wieder dem ursprünglichen Sender zuordnen und im internen Netz richtig zustellen kann.

3.1 Paket Filter

Der wirksamste Platz zum Filtern von Paketen ist der Rechner, der als Schnittstelle zwischen dem geschützten und dem öffentlichen Netz steht. In unserem Beispiel handelt es sich dabei um den Rechner `extor.firma.de`.

ipfwadm

Als Beispiel für einen *Packet Filter* wird das Programm **ipfwadm**[3] verwendet, da dies für Linux[4] am geeignetsten erscheint.

Der *Packet Filter* im Linux-Kern unterstützt das Filtern von ankommenden, abgehenden und weiterzuleitenden Paketen. Jeder dieser Filter besteht aus Filterregeln und Vorschriften. Trifft auf ein Paket eine Regel zu, so besagen die Vorschriften, wie mit ihm weiter zu verfahren ist. Der Algorithmus, der in den Filtern abläuft, kann wie folgt beschrieben werden:

1. Wende alle Filterregeln des gegebenen Filters auf das Paket an.
2. Die erste Regel, die zutrifft, bestimmt alle folgenden Aktionen:
   - Wende die zur Regel gehörenden Vorschriften auf das Paket an.
   - Erhöhe den Paket- und Bytezähler für diese Regel.
   - Protokolliere gegebenenfalls die Aktion im *Kernel-Logfile*[5].
3. Trifft keine der Regeln zu, wende die implizite Regel des Filters an.

Es gibt vier Vorschriften, von denen eine auf das Datagramm anzuwenden ist. Sie lauten:

Accept: Datagramm weiterleiten.

Deny: Datagramm verwerfen.

Reject: Datagramm verwerfen und gleichzeitig die ICMP-Mitteilung *Destination Unreachable* dem Sender schicken.

Masquerade: IP-*Masquerading* auf das Datagramm anwenden. Dies ist nur im Rahmen eines weiterleitenden Filters möglich.

[2] ein *Port* mit einer Nummer zwischen 1024 und 65535

[3] `http://www.xos.nl/linux/ipfwadm/`

[4] der Linux-Kern muß mit den „Networking options“ `IP firewalling` und `IP forwarding` übersetzt werden

[5] mittels **syslogd**

Die einzelnen Regeln werden durch die nachfolgenden Attribute spezifiziert, nach denen festgelegt wird, ob das gegebene Paket die Regel erfüllt oder nicht.

- Eine Menge von Sender- und Empfängeradressen. Die IP-Adressen werden als Paar, *adresse/maske* angegeben, wobei die *maske* entweder in gewohnter Schreibweise oder als natürliche Zahl angegeben werden kann[6]. Ergibt die Konjunktion von *maske* und IP-Adresse des Datagramms *adresse*, so gehört das Datagramm zu dieser Menge. Bei einer *maske* mit lauter Nullen werden alle IP-Adressen akzeptiert.
- Die Protokolle: TCP, UDP, ICMP oder gegebenenfalls alle drei.
- Die Nummern der *Ports* zusammen mit den zugehörigen Protokollen TCP oder UDP. Für jede Regel können bis zu zehn Sender- und Empfänger-*Ports* angegeben werden[7].
- Die durch ICMP erzeugten Meldungen.
- Die *Flags* SYN[8] und ACK im TCP-Segment. Dadurch kann der Aufbau von neuen TCP-Verbindungen in eine bestimmte Richtung unterbunden werden. SYN, ACK
- Die IP-Adresse der Netzschnittstelle, die das Datagramm angenommen hat oder über die es versendet wird.
- Die symbolische Bezeichnung der Netzschnittstelle, die das Datagramm angenommen hat oder über die es versendet wird.

3.1.1 Accounting

Ein nützliches Nebenprodukt des *Packet Filtering* ist das Zählen von Paketen und Bytes.

Beispiel. *Das* Accounting *am Rechner* `extor.firma.de` *kann wie folgt initiiert werden:*

```
extor:~# ipfwadm -A -a -S 0.0.0.0/0 -D 192.168.33.0/24 -V 172.24.233.253
```

Durch den Schalter **-a** *wird die Regel den* Accounting*-Regeln (Schalter* **-A***) im Kern hinzugefügt. Dies führt dazu, daß alle über die Netzschnittstelle* `172.24.233.253` *(Schalter* **-V***) in das Firmennetz* `192.168.33.0` *(Schalter* **-D***) geleiteten Daten gezählt werden.*

Sollen auch die nach außen gehenden Pakete gezählt werden, so muß eine zweite Regel mit vertauschten Schaltern **-D** (Destination) *und* **-S** (Source) *angegeben werden.*

Die Ausgabe der Accounting*-Regeln ergibt dann folgendes:*

```
extor:~# ipfwadm -A -l -n
IP accounting rules
```

[6] `192.168.33.0/255.255.255.0` entspricht der Angabe `192.168.33.0/24`

[7] einzeln oder als Bereich, z.B. `1024:65535` (mit Doppelpunkt statt Bindestrich)

[8] *Synchronize Sequence Numbers*, erstes, beim Aufbau einer Verbindung vom Sender geschicktes Segment, das dem Empfänger mitteilt, welche Sequenznummer das erste Datensegment des Senders haben wird (→ S.5). Der Empfänger antwortet mit ACK (*Acknowledgement*) und SYN, wodurch er den Vorschlag des Senders annimmt und ihm gleichzeitig mitteilt, mit welcher Sequenznummer seine Übertragung begonnen wird. Zu guter letzt schickt der Sender dem Empfänger ein Segment (ACK), mit dem er den Vorschlag des Empfängers bestätigt. Danach fängt die eigentliche Datenübertragung an. Diese Sequenz zu Beginn jeder Übertragung wird als *Three Way Handshake* bezeichnet.

```
pkts bytes  prot source              destination         ports
765K  581M  all  192.168.33.0/24     0.0.0.0/0           n/a
621K  569M  all  0.0.0.0/0           192.168.33.0/24     n/a
```

3.1.2 Konfiguration der Filter

Im Gegensatz zum *Accounting* kommt es bei den Filtern für ankommende, abgehende und weiterzuleitende Pakete auf die Reihenfolge der Regeln an. Mit Hilfe der Schalter **-i** und **-a** können daher neue Regeln dem Regelsatz des entsprechenden Filters voran- oder hintangestellt werden. Des weiteren muß angegeben werden, welche Vorschrift auf die Pakete angewendet werden soll.

Mit Hilfe des Schalters **-p** kann eine implizite Reaktion eingestellt werden, die angewendet wird, falls keine andere Regel des Filters auf das Paket zutrifft.

Aufgrund der Beachtung der Reihenfolge der Regeln und mit Hilfe der impliziten Reaktion kann das Verhalten des Filters von „alles verbieten“ bis „gezielt erlauben“ eingestellt werden.

Beispiel. *Zur Demonstration der Wirkung von Filtern soll auf* `extor.firma.de` *das Protokoll (Schalter* **-P***) UDP bei von außen kommenden Paketen herausgefiltert werden. Dazu werden zuerst alle Regeln des Filters für ankommende Pakete gelöscht und danach die implizite Reaktion auf* **accept** *gesetzt. Anschließend werden alle UDP-Datagramme, die von außen kommen (Schalter* **-I** *und* **-V***), abgeblockt.*

```
extor:~# ipfwadm -I -f
extor:~# ipfwadm -I -p accept
extor:~# ipfwadm -I -a reject -P udp -S 0.0.0.0/0 -D 192.168.33.0/24 -V 172.24.233.253
```

*Durch die Angabe der Netzschnittstelle (***-V 172.24.233.253***) werden nur die UDP-Datagramme, die von außen kommen, abgeblockt; Stationen aus dem Netz* `192.168.33.0` *können weiterhin mit* `extor.firma.de` *über UDP kommunizieren.*

Der Austausch von Routing*-Informationen zwischen den Rechnern* `extor.firma.de` *und* `verbinde.extern.de` *ist jedoch nicht mehr über RIP sondern gegebenenfalls nur noch über OSPF möglich. Eine weitere Folge ist, daß der DNS-*Server *auf* `extor.firma.de` *bei dieser Einstellung nicht mehr Daten mit externen DNS-*Servern *austauschen kann*[9]*.*

Dieses Manko könnte durch folgendes Kommando behoben werden:

```
extor:~# ipfwadm -I -i accept -P udp -S 0.0.0.0/0 -D 192.168.33.94 53
```

Port 53 *Dadurch wird der Austausch von DNS-Informationen über* Port *53 freigegeben. Durch den Schalter* **-i** *wird diese Angabe vor die anderen in den Regelsatz für ankommende Pakete gestellt. Dies ist wichtig, da sonst auch DNS-Datagramme generell abgelehnt würden.*

[9] der Kontakt erfolgt über UDP (max. Paketlänge 512 Byte, *Port* 53), bei Antworten mit mehr als 512 Byte wird TCP (*Port* 53) verwendet; Zoneninformationen (z.B. zwischen primärem und sekundärem *Server*) werden über TCP ausgetauscht

Die Konfiguration des *Packet Filters* kann zu unerwünschten Seiteneffekten führen, welche die Sicherheit des Netzes gefährden können. Hilfestellung bei der Einrichtung des Filters kann z.B. das Paketanalyseprogramm **tcpdump**[10] bieten.

Beispiel. *Den WWW-*Clients *aus dem Firmennetz soll der Zugang zu* Servern *im* Internet *ermöglicht werden. Dazu wird zuerst jeglicher Verkehr in und aus dem Firmennetz unterbunden.*

```
extor:~# ipfwadm -I -p deny
```

*Nun wird die Kommunikation mit einem beliebigen WWW-*Server *auf* Port *80 gestattet.*

```
extor:~# ipfwadm -I -a accept -P tcp -S 0.0.0.0/0 80 -D 192.168.33.0/24 (Regel 1)
extor:~# ipfwadm -I -a accept -P tcp -S 192.168.33.0/24 -D 0.0.0.0/0 80 (Regel 2)
```

Der Rechner `extor.firma.de` *akzeptiert jetzt Verbindungen aus dem internen Netz zum* Port *80 eines beliebigen externen* Servers *(Regel 2) und vom* Port *80 eines externen* Servers *an einen beliebigen* Port *eines internen Rechners (Regel 1).*

Es ist sicherlich nicht sinnvoll zuzulassen, daß sich externe Server *an einem beliebigen* Port *eines internen Rechners anmelden können. Abhilfe bietet die Einschränkung der internen* Ports *auf Werte zwischen 1024 und 65535. Die erste Regel wird dazu wie folgt geändert:*

```
extor:~# ipfwadm -I -a accept -P tcp -S 0.0.0.0/0 80 -D 192.168.33.0/24 1024:65535
```

Sollen nur die internen Benutzer eine TCP-Verbindung aufbauen dürfen, so kann durch

```
extor:~# ipfwadm -I -i accept -P tcp -k -S 0.0.0.0/0 -D 192.168.33.0/24
```

jeglicher Verbindungsaufbau von außen unterbunden werden. Eine wichtige Rolle spielt dabei der Schalter **-k**, *der diese Regel auf Datagramme einschränkt, die unter anderem das* Flag *ACK im Kopf des TCP-Segments gesetzt haben. Da das erste Datagramm, mit dem eine externe Station eine TCP-Verbindung zu einem internen Rechner aufbaut, dieses* Flag *nicht enthält, wird sie abgewiesen. Wird der Aufbau von einer internen Station initiiert, so antwortet die externe mit ACK und die Verbindung wird aufgebaut.*

Auch mit der zweiten Regel kann es Probleme geben, wenn eine externe Station vorgibt, zum Netz `192.168.33.0` *zu gehören*[11]*. Dies kann durch die nachfolgende Regel, bei der die Netzschnittstelle mit angegeben wird, verhindert werden.*

```
extor:~# ipfwadm -I -i deny -V 172.24.233.253 -S 192.168.33.0/24 -D 0.0.0.0/0
```

Jetzt werden alle Datagramme, die vorgeben aus dem internen Netz zu stammen, jedoch über `172.24.233.253` *von außen kommen, abgewiesen.*

3.2 Proxy Server

Während ein *Packet Filter* wie ein selektiver *Router* funktioniert und immer wenigstens einen Teil des Datenverkehrs weiterleitet, überträgt ein *Bastion Host* (Bastion, *Bastion Host*

[10] `http://ee.lbl.gov/`
[11] IP *Spoofing*

Bollwerk) keinerlei Daten.

Der *Bastion Host* wird durch eine stellvertretende Applikation (*Proxy Application* bzw. *Proxy Server*) realisiert. Diese funktioniert auf der Applikations-Schicht, wogegen der *Packet Filter* auf der Internet- und Transport-Schicht arbeitet.

Der *Bastion Host* wird sinnvollerweise auf dem Rechner installiert, der die Verbindung zwischen dem öffentlichen und dem geschützten Netz herstellt[12]. Dies kann auf zweierlei Art und Weise geschehen.

- Der Verbindungsrechner wird als *Bastion Host* eingerichtet. Im Firmennetz wäre dies der Rechner `extor.firma.de`.
- Der Verbindungsrechner fungiert als *Packet Filter*, der nur Datagramme weiterleitet, die für einen zweiten Rechner, der als *Bastion Host* eingerichtet ist, bestimmt sind. Diese Kombination könnte z.B. durch `extor.firma.de` als *Packet Filter* und `intor.firma.de` aber auch z.B. `antje.firma.de` als *Bastion Host* realisiert werden.

Da beide Möglichkeiten vom Prinzip her gleich sind, wird im folgenden nur die erste besprochen.

FWTK

Für Linux gibt es mehrere kostenlose Programme, mit denen ein *Bastion Host* eingerichtet werden kann. Eins der bekanntesten ist das *Firewall Toolkit* (FWTK) der Firma *Trusted Information Systems*[13]. Es setzt sich aus mehreren Programmen zusammen, die alle mit Hilfe einer einzigen Datei (`/usr/local/etc/netperm-table`) konfiguriert werden. Der Quelltext, geschrieben in C, ist jedermann zugänglich.

Der Entwurf von FWTK erfolgte unter der Prämisse, möglichst alle Sicherheitsrisiken auszuschließen.

- Die Läufe von Programmen im priviligierten Modus werden auf ein Minimum beschränkt,
- das Wurzelverzeichnis wird mittels **chroot** vor dem Start eines Programms z.B. nach `/usr/local/etc`[14] verschoben, wo ein Angreifer keine wichtigen Systemprogramme findet und damit auch keinen größeren Schaden anrichten kann.

Die FWTK-Programme können hinsichtlich ihrer Funktion in drei Kategorien eingeteilt werden:

1. Programme, welche die Zugangskontrolle zu den einzelnen TCP-*Ports* (*TCP Wrapper*) regeln.
2. Programme zur Überprüfung der Identität der Benutzer.
3. *Proxy Applications*, die den Datenverkehr zwischen äußerem und geschütztem Netz vermitteln.

[12] dazu muß bei Linux im Abschnitt „Networking options“ `IP forwarding` verboten werden
[13] http://www.tis.com/
[14] genannt *Locked Room Directory*; sollte nicht mit dem Wurzelverzeichnis für anonyme FTP-Benutzer übereinstimmen (→ S.61)

3.2.1 Konfiguration des Bastion Host

Da der *Bastion Host* den Kontrollpunkt zwischen dem geschützten und öffentlichen Netz darstellt, sollte er selbst so wenig wie möglich angreifbar sein.

- Es sind alle Benutzerkonten, bis auf das des Administrators, zu löschen.
- Die vorhandenen Dienste sind auf ein Minimum zu reduzieren.
- NIS und NFS sind auszuschalten.
- Die Weiterleitung von Datagrammen ist zu unterbinden (→ Fußnote S.162).

Beispiel. *Linux sollte nach der Durchführung der eben genannten Maßnahmen mit folgenden Prozessen auskommen:*

```
extor:~$ ps ax
  PID TTY STAT TIME COMMAND
    1  ?  S    0:05 init [3]
    2  ?  SW   0:00 (kflushd)
    3  ?  SW<  0:00 (kswapd)
   12  ?  S    0:03 update (bdflush)
   77  ?  S    0:00 /usr/sbin/klogd -c 1
   79  ?  S    0:12 /usr/sbin/syslogd
  116  ?  S    0:00 /usr/sbin/cron
  119  ?  S    0:03 /usr/sbin/inetd
   74  3  SW   0:00 (agetty)
11214  p1 S    0:00 -bash
11236  p1 R    0:00 ps ax
```

Die *Proxy Applications* für die Internetdienste werden durch **inetd** aufgerufen. Dazu muß die Konfigurationsdatei **/etc/inetd.conf** entsprechend angepaßt werden. Wie dies zu geschehen hat, wird weiter unten besprochen.

3.2.2 Konfigurationsdatei /usr/local/etc/netperm-table

Jeder Eintrag in der Konfigurationsdatei hat folgende Form[15]:

programme: *daten*

Der Parameter *programme* kann einen oder mehrere Programmnamen aus der Menge der FWTK-Programme enthalten. Die Namen werden durch Kommas getrennt angegeben. Die Konfiguration der angeführten Programme erfolgt anhand von Angaben, die das Programm aus denjenigen Zeilen entnimmt, die seinen Namen enthalten (s.u.).

Der Rest der Zeile (*daten*) enthält Angaben, die für alle vor dem Doppelpunkt stehenden Programme gemeinsam sind. Handelt es sich dabei um Adreßbereiche, so können bei ihrer Spezifikation Sonderzeichen (*Wildcards*) verwendet werden, also z.B. `192.168.33.*`

[15] es dürfen keine Fortsetzungszeilen verwendet werden (unter Verwendung von \), Kommentare fangen mit # an

oder `*.firma.de`. Es wird dabei empfohlen, immer IP-Adressen anstatt von Domänennamen zu verwenden, damit ein potentieller Angreifer nicht vortäuschen kann, er sei aus der Domäne des geschützten Netzbereichs[16].

3.2.3 Kontrolle des Zugriffs auf TCP-Dienste

netacl Das Programm **netacl** ist ein *TCP Wrapper*, ähnlich wie **tcpd** (→ S.61). Es ist im Gegensatz zu **tcpd** nur auf TCP beschränkt, da davon ausgegangen wird, daß UDP über den *Bastion Host* gänzlich unterbunden wird.

Beispiel. *Die auf Seite 61 angegebenen Zeilen der Datei* **inetd.conf** *könnten wie folgt angepaßt werden:*

```
ftp    stream tcp nowait root /usr/local/etc/netacl ftpd
telnet stream tcp nowait root /usr/local/etc/netacl telnetd
```

Die Angabe für `tftp` *entfällt, da UDP generell verboten ist.*

In der **netperm-table** sehen die Angaben zur Konfiguration der **netacl**-Dienste etwas anders als oben angegeben aus.

`netacl-`*dienst*`: permit-hosts` *stationen vorschriften*
`netacl-`*dienst*`: deny-hosts` *stationen*

Für *dienst* kann eine der Angaben aus **inetd.conf** stehen, also z.B. `ftpd`, `telnetd`, `rlogind`, `fingerd`, usw.

Der Parameter *stationen* enthält die IP-Adressen oder Domänennamen, die nach den oben genannten Regeln angegeben werden. Eine Station, die versucht einen der durch **netacl** „eingepackten“ Dienste anzufordern, wird mit diesen Angaben verglichen, die erste Spezifikation die zutrifft, bestimmt die Reaktion von **netacl**.

Handelt es sich dabei um eine Zeile mit `deny-hosts`, so wird die Verbindung abgebrochen und mittels **syslogd** der Abbruch im *Kernel-Logfile* protokolliert.

Enthält die Zeile `permit-hosts`, so bestimmen die *vorschriften*, wie weiter zu verfahren ist.

Folgende *vorschriften* stehen zur Auswahl:

`-user` *benutzer*	Gibt die UID (*User Identification*) oder den Namen des Benutzers an, unter dessen Zugangsberechtigung das Programm gestartet werden soll.
`-chroot` *verzeichnis*	Vor dem Start des bei `-exec` angegebenen Programms wird mittels **chroot** das Wurzelverzeichnis nach *verzeichnis* gelegt. Das zu startende Programm muß folglich unter dem neuen Wurzelverzeichnis liegen, da sich alle Pfadangaben auf dieses beziehen.
`-exec` *programm argumente*	Gibt das Programm mit seinen Übergabeparametern an. Diese Angabe darf bei Zeilen mit `permit-hosts` nicht fehlen und sie muß am Ende der Zeile stehen.

[16] DNS *Spoofing*

Beispiel. *Um den internen Stationen die Benutzung des Dienstes* **finger** *zu gestatten und den externen stattdessen eine Nachricht darüber zu schicken, daß der Dienst für sie blockiert ist, müßten folgende zwei Zeilen in der Datei* **netperm-table** *stehen:*

```
netacl-fingerd: permit-hosts 192.168.33.* -exec /usr/sbin/fingerd
netacl-fingerd: permit-hosts * -exec /bin/cat /usr/local/etc/finger.txt
```

3.2.4 Authentifizierung der Benutzer

Das FWTK-Paket enthält das Programm **authsrv**, das fünf unterschiedliche Methoden zur Identifizierung von Benutzern zur Verfügung stellt. **authsrv**

Die am häufigsten verwendete ist S/Key[17] der Firma Bellcore.

Bei S/Key werden Passwörter bei jedem Verbindungsaufbau neu generiert[18], so daß das Abhören von Passwörtern sinnlos wird. Das geheime Benutzerpasswort ist dabei nur dem Benutzer selbst bekannt. Um es **authsrv** mitzuteilen, haben die *Proxy Applications* für **telnet** und **rlogin** ein **passwd**-Kommando integriert, das dem Benutzer die Eingabe eines neuen oder die Änderung eines bestehenden Passwortes gestattet[19]. Dies sollte aus Sicherheitsgründen jedoch nur aus dem geschützten Netz heraus erfolgen[20].

Das geheime Passwort dient als Grundlage zur Erzeugung der Einwegpasswörter. Dies geschieht mit Hilfe einer sogenannten *Hash*-Funktion (f). Sie ist öffentlich und einfach anzuwenden, *Hash Function*

$$y = f(x),$$

es ist jedoch aus Aufwandsgründen kaum möglich, die inverse Funktion zu ihr zu bilden,

$$x = f^{-1}(y).$$

Die für den S/Key-Algorithmus gewählte *Hash*-Funktion kann $2^{64} \approx 10^{19}$ unterschiedliche Werte[21] y erzeugen.

Zur Berechnung von f wird der MD4-Algorithmus[22] verwendet. MD4 erzeugt aus einer beliebig langen Bitsequenz eine 16 Byte lange Ausgabe. Er ist schnell, und da er bisher noch nicht geknackt werden konnte, gilt er auch als sicher. MD4

Die gewählte *Hash*-Funktion erwartet 8 Byte lange Eingaben, aus denen sie wiederum 8 Byte lange Ausgaben erzeugt.

Da die geheimen Passwörter länger als 8 Byte sein können, erfolgt vor der ersten Anwendung von f ein Anpassungsschritt. Dazu wird zuerst an das geheime Passwort ein Text (genannt *Seed*) angehängt, den der *Client* vom *Server*[23] erhält. Darauf wird der MD4-Algorithmus angewendet und das Ergebnis durch paarweise Antivalenzbildung von 16 auf 8 Byte reduziert. Das Ergebnis s dieser Operationen dient dann als Eingabe für f.

[17] `ftp://ftp.bellcore.com/pub/nmh/skey`
[18] sogenannte *one-time passwords* (Einwegpasswörter bzw. Einmalpasswörter)
[19] Eingabe des Benutzernamens und Passwortes, Wiederholung der Passworteingabe
[20] oder direkt am Authentifizierungs-*Server* eingegeben werden
[21] 8 Byte lange Einwegpasswörter
[22] *Message Digest Function #4*, entwickelt von Ronald Rivest (einem der Entwickler des RSA-Algorithmus) und veröffentlicht unter RFC 1320 (MD5 unter RFC 1321)
[23] es handelt sich um einen lesbaren Text

Das erste Einwegpasswort wird erzeugt, indem auf s die *Hash*-Funktion N-mal angewendet wird,

$$p_0 = f^N(s),$$

das nächste, indem die *Hash*-Funktion nur noch $N-1$-mal angewendet wird,

$$p_1 = f^{N-1}(s),$$

usw. Allgemein gilt

$$p_i = f^{N-i}(s).$$

Hat ein Lauscher das Einwegpasswort p_i abgehört, so ist er nicht in der Lage das nächste in der Folge (p_{i+1}) zu bestimmen, da er dazu f invertieren müßte[24].

Angenommen, das letzte Einwegpasswort hatte die Sequenznummer i. Meldet sich jetzt ein Benutzer erneut an, so erhält er vom *Server* zusammen mit dem Text (*Seed*) die neue Sequenznummer $i+1$. Er gibt diese zusammen mit dem Text ein, worauf er aufgefordert wird, das geheime Passwort anzugeben. Der *Client* erzeugt nach Eingabe des Passwortes p_{i+1} und schickt es an den *Server*. Der *Server* speichert p_{i+1} und wendet einmal die *Hash*-Funktion auf p_{i+1} an.

$$p_i = f(f^{N-i-1}(s)) = f(p_{i+1})$$

Er vergleicht nun p_i mit dem Passwort in seiner Passwortdatei. Stimmen die Werte überein, so wird der Benutzer zugelassen, p_i durch p_{i+1} überschrieben und die Sequenznummer auf $i+2$ erhöht. Beim nächsten Anmelden des *Clients* wird f einmal weniger auf s angewendet, wodurch p_{i+2} entsteht. Nun wird

$$p_{i+1} = f(f^{N-i-2}(s)) = f(p_{i+2})$$

aus p_{i+2} ermittelt und mit dem Eintrag in der Passwortdatei verglichen. Diese Schritte wiederholen sich bei jedem Anmelden so lange, bis $N-i$ so klein geworden ist, daß ein erneutes Initialisieren der Sequenz notwendig wird.

Außer dem eben beschriebenen Algorithmus, der auf einer Idee von Leslie Lamport aus dem Jahr 1981 basiert, stehen noch SecurID (Fa. Security Dynamics), SNK (Fa. Digital Pathways), Silver Card (Fa. Enigma Logic) und die nicht zu empfehlende Möglichkeit, einfache Textpasswörter zu verwenden, zur Verfügung.

Die Auswahl der Authentifizierungsprotokolle erfolgt vor dem Übersetzen von **authsrv** in der Datei **auth.h**.

Die Funktion der Authentifizierung wird am besten auf dem *Bastion Host* realisiert. Dazu muß zunächst **authsrv** ein freier *Port* zugeordnet und in der Datei **/etc/services** zugewiesen werden.

Beispiel. *In* **/etc/services** *wird dazu z.B. folgende Zeile hinzugefügt:*

```
authsrv      3456/tcp
```

Meldet sich ein Benutzer am System an, so wird seine Authentizität durch **authsrv** überprüft. Dazu muß das Programm durch **inetd** gestartet werden.

[24] das nächste wird ja dadurch erzeugt, daß f einmal weniger auf s angewendet wird

Beispiel. *In* **/etc/inetd.conf** *ist dazu folgende Zeile hinzuzufügen:*

```
authsrv stream tcp nowait root /usr/local/etc/authsrv authsrv
```

Für die Benutzer wird auf dem *Server* eine Datenbasis erzeugt, die unter anderem für jeden Benutzer seinen Namen, seine Gruppenzugehörigkeit, das Authentifizierungsprotokoll, seinen bürgerlichen Namen und den Zeitpunkt seiner letzten erfolgreichen Anmeldung am System, enthält. Des weiteren besteht die Möglichkeit, unterschiedliche Zugangsgruppen zu bilden, die jeweils von authorisierten Systemverwaltern betreut werden. Dies hat den Vorteil, daß ein Authentifizierungs-*Server* von unterschiedlichen Abteilungen einer Organisation gemeinsam genutzt werden kann, wobei jede Abteilung einen eigenen Systemverwalter haben kann, der die Benutzer seiner Zugangsgruppe unabhängig von den anderen Gruppen verwaltet.

Wird **authsrv** auf dem *Server* durch einen Benutzer mit der UID 0 (**root**) gestartet, kann die Administration mit einfachen Befehlen interaktiv erfolgen.

Beispiel. *Die Initialisierung der Datenbasis*[25] *erfolgt mit dem Eintrag für den Systemadministrator. Dies kann nach dem Aufruf des Kommandos* **authsrv** *direkt am* Server *oder mit* **authmgr** *über das Netz, erfolgen.* **authmgr**

```
extor:~# authsrv
-administrator mode-
authsrv# list
Report for users in database
user    group    longname        flags  proto   last
----    -----    --------          ---  -----   ----
authsrv# adduser admin "Authsrv admin"
ok - user added initially disabled
authsrv# ena admin
enabled
authsrv# superwiz admin
set wizard
authsrv# proto admin Skey
changed
authsrv# pass admin "GePassWort"
Secret key changed
authsrv# list
Report for users in database
user    group    longname        flags  proto   last
----    -----    --------          ---  -----   ----
admin            Authsrv admin     y W  Skey    never
authsrv# quit
```

Der zunächst leeren Datenbasis wird der Benutzer **admin** *hinzugefügt (***adduser***), aktiviert (***ena***) und zum globalen Systemverwalter (***superwiz***) bestimmt. Danach wird ihm das Authentifizierungsprotokoll S/key zugeordnet (***proto***) und sein geheimes Passwort (***pass***) eingegeben*[26]*.*

[25] sie ist als *Hash Table* des Typs NDBM (→ S.49) realisiert

[26] erfolgt die Einrichtung über **authmgr**, so wird kein Echo auf dem Bildschirm erzeugt

authdump **authload**

Als weitere Programme stellt FWTK **authdump** und **authload** zur Verfügung, mit deren Hilfe die Datenbasis verwaltet werden kann. Um z.B. regelmäßig eine Sicherungskopie der Datenbasis zu machen, kann **authdump** zusammen mit **crontab** verwendet werden. Im Falle der Zerstörung der Datenbasis oder dem irrtümlichen Löschen von Einträgen hat man dann ihre ASCII-Kopie, aus der **authload** wieder die ursprüngliche Datenbasis herstellen kann.

Die Authentifizierung der Benutzer unterstützen alle *Proxy Applications*, die FWTK zur Verfügung stellt. Die Benutzung dieser Mechanismen wird nachfolgend zusammen mit der Konfiguration der *Proxy Applications* für die unterschiedlichen Netzdienste gezeigt.

3.2.5 Stellvertretende Applikationen

Die Aufgabe des *Bastion Host* ist es, eine direkte Verbindung zwischen *Client* und *Server* zu unterbinden. Daher gibt es für Netzdienste, die sich des Protokolls TCP bedienen, stellvertretende Applikationen, die als Vermittler zwischen den beiden auftreten. Das Paket FWTK enthält solche *Proxy Applications* für Telnet, Rlogin, FTP, Gopher, HTTP, SMTP, X Window und eine weitere, die für unterschiedliche Zwecke (z.B. NNTP) verwendet werden kann.

Ein externer *Client*, der mit einem Rechner im geschützten Netz kommunizieren möchte, muß sich an die entsprechende stellvertretende Applikation auf dem *Bastion Host* wenden. Dieser muß er sich zuerst ausweisen und dann die gewünschte Verbindung anfordern. Ist alles in Ordnung, so verbindet sich die stellvertretende Applikation mit der internen Station und übergibt die Daten von einer TCP-Verbindung an die andere.

Beispiel. *Möchte ein Benutzer von außen eine Telnet-Verbindung mit* `antje.firma.de` *aufbauen, so verbindet er sich zuerst mit dem* Bastion Host `extor.firma.de`*,*

```
alien:~$ telnet extor.firma.de
```

auf dem nach der Eingabe von Benutzernamen und Passwort die Eingabezeile der stellvertretenden Applikation `tn-gw` *erscheint. In dieser gibt der Benutzer den Rechner an, mit dem er in Verbindung treten will.*

```
tn-gw->c antje.firma.de
```

Die Proxy Application *überprüft nun anhand der Konfiguration, ob der Benutzer berechtigt ist, sich mit* `antje.firma.de` *zu verbinden, protokolliert dies mittels* **syslogd** *und vermittelt gegebenenfalls die Verbindung.*

3.2.5.1 Stellvertretende Applikationen für Telnet und Rlogin

Die Anmeldung von Benutzern am *Bastion Host* sollte auf administrative Aufgaben beschränkt bleiben. Abhängig von den lokalen Umständen bieten sich dazu drei Möglichkeiten an.

1. Die Anmeldung ist nur direkt am *Bastion Host* über die lokale Tastatur möglich. Dies ist zwar die sicherste aber auch die unbequemste Möglichkeit. Sitzt der Systemverwalter nicht in der Nähe des Rechners, der als *Bastion Host* fungiert, kann es außerdem bei einer Störung zu zu langen Antwortzeiten kommen.

2. Die stellvertretende Applikation **tn-gw** wird auf den Standard-*Port* von Telnet (23) gesetzt und dem Telnet-*Daemon* **telnetd** wird ein anderer *Port* zugewiesen. An diesem können sich dann die Administratoren von kontrollierten Stationen aus dem Netz anmelden. Die gleiche Vorgehensweise erfolgt bei Rlogin. **tn-gw**

   Der Nachteil dieser Variante ist, daß Applikationen, die Telnet oder Rlogin verwenden, Probleme mit der Verbindung an die ephemeren *Ports* haben können.

3. Mit Hilfe des *Wrappers* **netacl** wird unterschieden, ob die Aufforderung zur Verbindung vom *Bastion Host* oder über das Netz kommt. Es ist dann möglich, sich über die stellvertretenden Applikationen **tn-gw** bzw. **rlogin-gw** anzumelden. **rlogin-gw**
   Beispiel. *Der Zugang zum* Bastion Host *wird in der* **netperm-table** *durch folgende Zeilen geregelt.*

```
netacl-telnetd:  permit-hosts 127.0.0.1 192.168.33.94 -exec /usr/sbin/telnetd
netacl-telnetd:  permit-hosts * -exec /usr/local/etc/tn-gw
netacl-rlogind:  permit-hosts 127.0.0.1 192.168.33.94 -exec /usr/sbin/rlogind
netacl-rlogind:  permit-hosts * -exec /usr/local/etc/rlogin-gw
```

   In der Datei **/etc/inetd.conf** *sind dann noch die Zeilen für Telnet und Rlogin anzupassen.*

```
telnet stream tcp nowait root /usr/local/etc/netacl telnetd
rlogin stream tcp nowait root /usr/local/etc/netacl rlogind
```

   Der Benutzer hat nun die Möglichkeit, sich aus dem Netz (`*`*) anzumelden, wodurch die stellvertretende Applikation gestartet wird. Nach der Eingabe von „***c localhost***“ werden dann die* Daemonen **telnetd** *oder* **rlogind** *aufgerufen. Dadurch erhält aber auch jeder Benutzer, der die* Proxy Applications *für Telnet und Rlogin benutzen darf, die Möglichkeit, unerlaubt in den* Bastion Host *einzudringen. Es ist daher wichtig, daß die Anmeldung am* Bastion Host *durch einen Authentifizierungsmechanismus geschützt wird.*

   Zur Konfiguration von **tn-gw** *könnten folgende Zeilen in der* **netperm-table** *hinzugefügt werden:*

```
tn-gw:  welcome-msg /usr/local/etc/tn-welcome.txt
tn-gw:  denial-msg /usr/local/etc/tn-deny.txt
tn-gw:  help-msg /usr/local/etc/tn-help.txt
tn-gw:  timeout 1200
tn-gw:  authsrv localhost 3456
tn-gw:  permit-hosts 192.168.33.* -passok -xok
tn-gw:  permit-hosts * -auth -dest !127.0.0.1 -dest !192.168.33.94 -dest !172.24.233.253
```

   Die ersten drei Zeilen geben den Pfad zu Dateien an, deren Inhalt zur Begrüßung des Benutzers, bei seiner Ablehnung und als Hilfe, ausgegeben wird.

   Die nächste Zeile (`timeout`*) stellt das Intervall ein, nach dem bei fehlendem Datenverkehr die stellvertretende Applikation die Verbindung abbricht.*

   Danach folgt ein Verweis auf den Authentifizierungsserver. Dieser läuft lokal auf Port *3456.*

Die letzten beiden Zeilen beschreiben die Bedingungen für einen Verbindungsaufbau.

Die Benutzer aus dem geschützten Netz können die stellvertretende Applikation **tn-gw** *ohne die Eingabe eines Passwortes aufrufen. Desweiteren dürfen sie ihr geheimes Passwort in der Datenbasis des Authentifizierungsservers (*`-passok`*) ändern und eine stellvertretende Applikation für X Window aufrufen (*`-xok`*), wodurch es möglich wird, die Ausgaben von entfernten X* Clients *durch den lokalen X* Server *darzustellen.*

Die letzte Zeile gibt an, daß sich Benutzer aus dem externen Netz authentifizieren müssen (`-auth`*) und daß ihnen der Zugang zum* Bastion Host *verwehrt ist (die Ausrufezeichen negieren die IP-Adressen und schließen sie dadurch aus).*

Durch die angegebenen Regeln wird somit der Zugang zum Bastion Host *für lokale Benutzer mittels* **tn-gw** *gestattet, wogegen er für externe verboten wird.*

3.2.5.2 Stellvertretende Applikation für FTP

ftp-gw

Im Gegensatz zum **tn-gw** erlaubt die stellvertretende Applikation für FTP genauere Angaben zu den erlaubten bzw. verbotenen Aktionen.

Beispiel.

```
ftp-gw:  welcome-msg /usr/local/etc/ftp-welcome.txt
ftp-gw:  denial-msg /usr/local/etc/ftp-deny.txt
ftp-gw:  help-msg /usr/local/etc/ftp-help.txt
ftp-gw:  timeout 540
ftp-gw:  authsrv localhost 3456
ftp-gw:  permit-hosts 192.168.33.* -log { retr stor } -auth { stor }
ftp-gw:  permit-hosts * -authall
```

Die ersten fünf Zeilen entsprechen denjenigen aus dem vorhergehenden Beispiel.

Bei den Zugangsberechtigungen kann zusätzlich angegeben werden, welche der Aktionen durch **syslogd** *registriert (*`-log`*) und für welche eine Authentifizierung notwendig ist (*`-auth`*).*

*Bei Benutzern aus dem geschützten Netz werden sowohl Speicher- als auch Lesevorgänge protokolliert, wobei für das Speichern von Dateien eine Authentifizierung notwendig ist. Alle anderen Benutzer haben Zugriff auf den FTP-*Server*, wollen sie jedoch irgendeine Operation ausführen (*`retr`*, *`stor`*), so müssen sie sich zuvor authentifizieren.*

Bei der im Beispiel angegebenen Konfiguration hätten anonyme FTP-Benutzer keine Möglichkeit, auf den FTP-*Server*[27] zuzugreifen. Soll ein Zugriff auf den anonymen FTP-*Server* auf dem *Bastion Host* möglich sein, so ist es unumgänglich, ihn nur über den *Wrapper* `netacl` zuzulassen.

Beispiel.

```
netacl-ftpd:  permit-hosts 192.168.33.* -exec /usr/sbin/ftpd
```

[27] dieser sollte sinnvollerweise auf dem *Bastion Host* laufen

```
netacl-ftpd:  permit-hosts unknown -exec /bin/cat /usr/local/etc/ftp-deny.txt
netacl-ftpd:  permit-hosts * -chroot /net/ftp -exec /usr/sbin/ftpd
```

*Durch die letzte Zeile wird erreicht, daß für alle Verbindungen (nicht nur für anonyme Benutzer) das Wurzelverzeichnis vor dem Start des FTP-*Servers *nach* **/net/ftp** *verschoben wird. Dadurch kann die Fehlerwahrscheinlichkeit wesentlich reduziert werden.*

Wie bei Telnet bzw. Rlogin tritt auch hier wieder eine Kollision mit dem Standardport von FTP (21) auf. Es können ja nicht die stellvertretende Applikation und FTP über den gleichen *Port* angesprochen werden.

Um dieses Problem zu lösen, enthält das FWTK-Paket einen modifizierten FTP-*Daemon*, der nach der Eingabe des Benutzernamens[28] in der Lage ist, selbst die *Proxy Application* `ftp-gw` zu starten.

Beispiel. *Möchte sich ein externer Benutzer mit einem internen Rechner via FTP verbinden, so muß er sich zuerst am* Bastion Host *anmelden.*

```
alien:~$ ftp extor.firma.de
```

Gibt er dann z.B.

```
ftp>user kerstin@antje
```

ein, so wird die stellvertretende Applikation **ftp-gw** *gestartet, die ihn (falls es die Konfiguration erlaubt) mit dem Rechner* `antje.firma.de` *verbindet.*

3.2.5.3 Stellvertretende Applikation für HTTP

Das FWTK-Programmpaket enthält auch eine stellvertretende Applikation für HTTP und Gopher. Diese Funktion kann zwar auch z.B. von Apache (→ S.146) übernommen werden, der Vorteil der *Proxy Application* **http-gw** liegt jedoch in ihrer Einfachheit, woraus sich eine geringere Fehleranfälligkeit ergibt. **http-gw**

Da die aktuellen *Browser*-Versionen auf einfache Art und Weise erlauben, ein *Proxy Gateway* einzustellen, ist der Zugriff auf das WWW auch über einen *Bastion Host* vollkommen transparent und problemlos.

Beispiel. *Die Installation von* **http-gw** *erfolgt auf dem* Bastion Host *auf dem Standardport 80 für HTTP und gegebenenfalls auf* Port *70 für Gopher.*

Der **netperm-table** *sind folgende Zeilen hinzuzufügen:*

```
http-gw:  timeout 1200
http-gw:  deny-hosts unknown
http-gw:  permit-hosts 192.168.33.*
http-gw:  permit-hosts * -httpd www.firma.de
http-gw:  forward /pub/* -protocol ftp -tohost ftp.firma.de
```

Für Clients *aus dem internen Netz gibt es keinerlei Einschränkungen, externe Benutzer landen auf dem Firmenserver.*

Die letzte Zeile ermöglicht **http-gw** *die Weiterleitung von Anfragen der Art* `ftp://extor.firma.de/pub/...` *auf den FTP-*Server `ftp.firma.de`.

[28] in der Form *benutzer@rechnername*

3.2.5.4 Stellvertretende Applikation für SMTP

Obwohl z.B. **sendmail** selbst in der Lage ist, als stellvertretende Applikation zu arbeiten, wird es nur zum Versenden von Nachrichten jedoch nicht für ihren Empfang, verwendet. Der Grund dafür liegt in seiner großen Komplexität, die es unmöglich macht, alle Sicherheitsmängel von vornherein auszuschließen. Ein solches Sicherheitsloch nutzte z.B. auch der *Cracker*, der mit seinem „Internetwurm“ 1988 die meisten amerikanischen Rechner lahmlegte.

Die Gefahr von **sendmail** liegt auch darin, daß es mit den Rechten des Systemadministrators läuft und Zugang zum gesamten Dateisystem hat.

Das FWTK-Paket enthält daher zwei Programme, die eine Art *Wrapper* um **sendmail** herum bilden.

smap Das Programm **smap** nimmt *e-mails* entgegen und speichert sie in einem sogenannten *spool directory*. Dabei läuft es in einem unprivilegierten Modus und verlegt außerdem das Wurzelverzeichnis in das *spool directory*.

smapd Der *Daemon* **smapd** entnimmt angekommene Briefe aus dem *spool directory* und übergibt sie zur Weiterleitung an **sendmail**, das dazu nicht mehr im priviligierten Modus des Systemadministrators laufen muß.

Damit der Empfang von *e-mail* von **smap** übernommen werden kann, ist folgende Zeile in **/etc/inetd.conf** einzutragen:

```
smtp stream tcp nowait mail /usr/local/etc/smap smap
```

Wie man der Zeile entnehmen kann, ist der Benutzer nicht mehr **root**, sondern der unpriviligierte Benutzer **mail**.

Der zweite Teil des *Wrappers* für *Sendmail* ist der *Daemon* **smapd**[29]. Seine Aufgabe besteht darin, die *e-mails* in regelmäßigen Abständen aus dem *spool directory* zu entnehmen und **sendmail** zur Weiterleitung zu übergeben.

Beispiel. *Die Konfiguration von* **smap** *und* **smapd** *in der* **netperm-table** *kann z.B. wie folgt aussehen:*

```
smap, smapd: userid mail
smap, smapd: directory /var/spool/mail
smapd:       executable /usr/local/etc/smapd
smapd:       wakeup 180
smap:        timeout 3600
smap:        maxrecip 1000
smap:        maxbytes 2097152
```

Wie man an den ersten beiden Zeilen sehen kann, ist es möglich, daten *für mehrere* programme *gemeinsam anzugeben* (→ S.163).

Die erste Zeile gibt den Benutzernamen an, mit dessen Rechten die Programme gestartet werden, auf der nächsten Zeile steht der vollständige Pfad zum spool directory.

Die nächste Zeile enthält den vollständigen Pfad zum **smapd**.

[29] der z.B. aus **rc.local** gestartet wird

Durch `wakeup 180` *(Sekunden) wird das Intervall festgelegt, in dem* **smapd** *aktiviert wird.*

Die restlichen drei Zeilen enthalten Angaben für **smap**.

Mit `timout` *wird angegeben, wie lange die Verbindung aufrechterhalten werden soll, wenn kein Datenverkehr erfolgt.*

Die letzten zwei Zeilen geben Maximalwerte für die Anzahl der Adressaten einer Nachricht (`maxrecip`*) bzw. für die Länge einer Nachricht (*`maxbytes`*) an. Diese Einschränkungen sollen ein Verstopfen von* **sendmail** *verhindern helfen.*

3.3 Verschlüsselung der elektronischen Post

Im folgenden soll *Pretty Good Privacy* (PGP) rudimentär untersucht werden. Dazu wird kurz eine Möglichkeit, die dem Benutzer mit PGP zur Verfügung steht, beschrieben und anschließend auf das zentrale Verschlüsselungsverfahren dieser Kombination näher eingegangen.

RSA, in Kombination mit MD5 und IDEA, sind jedoch nicht die einzigen Verfahren, die PGP verwendet. Es würde allerdings den Rahmen dieser Einführung sprengen, auf alle Möglichkeiten, die dem Benutzer zur Verfügung stehen (DH/DSS statt RSA; CAST, IDEA oder 3DES zur Verschlüsselung und SHA als Ersatz für MD5), einzugehen. Ich werde mich folglich im nächsten Abschnitt auf die Erklärung des RSA-Algorithmus beschränken.

Der MD5-Algorithmus (*Message-Digest*) bildet eine 128-Bit lange Quersumme aus einer beliebig langen Nachricht. Der RSA-Algorithmus wird auf diese Quersumme angewendet, um die digitale Unterschrift zu erzeugen.

Der IDEA-Algorithmus (*International Data Encryption Algorithm*) verwendet für jede E-Mail einen neuen, zufällig erzeugten 128-Bit Schlüssel, mit dem die Nachricht chiffriert wird.

Und jetzt zu den einzelnen Punkten in Tabelle 3.1.

Digitale Unterschrift

Die digitale Unterschrift wird wie folgt eingesetzt:

1. Der Sender erzeugt eine Nachricht.
2. Mit Hilfe von MD5 wird eine 128-Bit lange Quersumme der Nachricht erzeugt.
3. Die Quersumme wird mit dem RSA-Algorithmus verschlüsselt, wozu der private Schlüssel des Senders benutzt wird. Das Ergebnis wird der Nachricht vorangestellt.
4. Der Empfänger entschlüsselt mit dem öffentlichen Schlüssel des Senders mit Hilfe von RSA die digitale Unterschrift und erhält so wieder die Quersumme.
5. Der Empfänger erzeugt mit MD5 eine neue Quersumme und vergleicht beide. Sind sie identisch, so ist die Nachricht authentisch.

Funktion	Algorithmus	Beschreibung
Chiffrierung der Nachricht	**IDEA, RSA**	Die Nachricht wird mit dem IDEA-Algorithmus verschlüsselt. Dazu wird ein Einwegschlüssel (*one time session key*) vom Sender erzeugt. Dieser Einwegschlüssel wird mit dem öffentlichen Schlüssel des Empfängers mit Hilfe des RSA-Algorithmus verschlüsselt und der Nachricht vorangestellt.
Digitale Unterschrift	**MD5, RSA**	Mit MD5 wird eine Quersumme (*hash code*) der Nachricht erzeugt. Diese wird dann mit dem privaten Schlüssel des Senders mit Hilfe des RSA-Algorithmus verschlüsselt und der Nachricht vorangestellt.
Kompression	**ZIP**	Die Nachricht kann vor dem Versenden mit ZIP komprimiert werden.
E-Mail Kompatibilität	**base64**	Eine verschlüsselte Nachricht kann mit base64 nach ASCII konvertiert werden.

Tabelle 3.1: Zusammenfassung der PGP-Dienste

Vertraulichkeit

Vertraulichkeit wird durch die Verschlüsselung der Nachricht vor ihrer Übertragung hergestellt. Dazu wird die Nachricht mit Hilfe des IDEA-Algorithmus verschlüsselt. Der dazu notwendige konventionelle Schlüssel wird nur einmal verwendet. Für jede Nachricht wird also ein neuer, zufällig erzeugter 128-Bit Schlüssel benötigt. Damit die Nachricht beim Empfänger wieder entschlüsselt werden kann, muß der Schlüssel mit übertragen werden. Um ihn vor Angriffen zu schützen, wird er vorher mit dem öffentlichen Schlüssel des Empfängers chiffriert.

Es ergibt sich folgender Ablauf:

1. Der Sender erzeugt die Nachricht und einen zufälligen 128-Bit Schlüssel, der nur für diese Nachricht verwendet wird (Einwegschlüssel).
2. Die Nachricht wird verschlüsselt, wozu der IDEA-Algorithmus mit dem Einwegschlüssel verwendet wird.
3. Der Einwegschlüssel wird mit dem öffentlichen Schlüssel des Empfängers mit dem RSA-Algorithmus verschlüsselt und der Nachricht vorangestellt.
4. Der Empfänger verwendet RSA und seinen privaten Schlüssel, um den Einwegschlüssel zu dechiffrieren.
5. Mit Hilfe des Einwegschlüssels entschlüsselt er dann die Nachricht.

Vertraulichkeit und digitale Unterschrift

Beide Dienste können natürlich auch zusammen verwendet werden. Dazu wird zunächst eine digitale Unterschrift mit dem privaten Schlüssel des Senders erzeugt und der Nachricht vorangestellt. Danach wird die Nachricht zusammen mit der digitalen Unterschrift mit dem IDEA-Algorithmus unter Verwendung des Einwegschlüssels chiffriert. Der Einwegschlüssel wird dann mit dem RSA-Algorithmus und dem öffentlichen Schlüssel des Empfängers chiffriert und der Nachricht vorangestellt.

Kompression

PGP komprimiert die Nachricht automatisch vor der Verschlüsselung, nachdem die digitale Unterschrift hinzugefügt wurde. Der Hauptgrund, warum die Quersumme und die digitale Unterschrift vor der Kompression gebildet werden müssen ist das nichtdeterministische Ergebnis des Kompressionsalgorithmus. Je nach Implementierung können nämlich Unterschiede bei der Kompression auftreten. Für die Dekompression sind diese jedoch irrelevant, man erhält immer das korrekte Ergebnis. Würde man folglich die Quersumme und digitale Unterschrift nach der Kompression bilden, würde dies PGP auf einen einzigen Kompressionsalgorithmus beschränken, da sonst kein eindeutiges Ergebnis bei der Dekompression zu erzielen wäre.

Wird die komprimierte Nachricht verschlüsselt, so erhöht das die kryptographische Sicherheit, da die komprimierte Nachricht wesentlich weniger Redundanzen enthält als die ursprüngliche Textnachricht.

E-Mail Verträglichkeit

Wenn PGP verwendet wird, so ist wenigstens ein Teil der Nachricht verschlüsselt und damit ein 8-Bit Datenstrom, der nicht mehr RFC 822 (→ S.48) entspricht. Da nicht alle elektronischen Postsysteme dies tolerieren, bietet PGP eine Konvertierung nach MIME mit base64 an.

Allein zu PGP sind einige Bücher erschienen, ganz zu schweigen von den angeführten Algorithmen. Zu PGP sind insbesondere [17] und [22] zu empfehlen, die Algorithmen werden z.B. in [16] oder [18] behandelt. Es ist also auf diese Bücher verwiesen, wenn sich der Leser tiefer in die Materie einarbeiten möchte.

Da der RSA-Algorithmus derjenige ist, der dem ganzen Verschlüsselungsverfahren die notwendige Sicherheit gibt, soll er im folgenden genauer betrachtet werden. Da die verwendete Mathematik nicht vorausgesetzt werden kann, werden zuerst die relevanten Gesetze aus der Zahlentheorie angegeben und bewiesen.

3.4 Der Algorithmus von Rivest-Shamir-Adleman (RSA)

Die Höhere Mathematik des Ingenieurstudiums beinhaltet in der Regel keine Zahlentheorie. Der Informatiker muß sich damit auseinandersetzen und ist daher, zumindest auf diesem Gebiet, dem Ingenieur voraus. Um diese Diskrepanz in puncto modulare Arithmetik auszugleichen, folgen anschließend einige Betrachtungen zu diesem Thema. Die Absicht ist, dem Leser den RSA-Algorithmus zum Chiffrieren und Dechiffrieren von

Nachrichten mit Hilfe von öffentlichen und privaten Schlüsseln näher zu bringen und ihn für ihn verständlich zu machen.

3.4.1 Etwas Zahlentheorie

Die Stärke des RSA-Verfahrens liegt in der Schwierigkeit, große Zahlen n in ihre Primfaktoren (in diesem Fall in ihre beiden Primfaktoren p und q) zu zerlegen. Es gibt zur Zeit keinen vernünftigen Algorithmus, der in der Lage wäre, Zahlen mit mehreren hundert Stellen in ihre beiden Primfaktoren zu zerlegen. Der beste bekannte Algorithmus schafft die Zerlegung einer Zahl n in einer Zeit proportional zu $\mathrm{L}(n) = \exp\left[\sqrt{\ln n \cdot \ln(\ln n)}\right]$.

Dies hat zur Folge, daß mit heutiger Technik eine 100-stellige Zahl in ungefähr zwei Wochen, eine 150-stellige in zirka einem Jahr und eine 200-stellige, eine Rechenleistung von 10^{12} Operationen pro Sekunde vorausgesetzt, in tausend Jahren zerlegt werden könnte.

Da die Primzahlen eine wichtige Rolle beim RSA-Verfahren spielen, soll mit ihnen begonnen werden. Zuerst aber eine Definition.

Definition 1. Es seien $a, d \in \mathbb{Z}$, wobei $d \neq 0$ gilt. Dann ist d ein Divisor von a (oder a ein Vielfaches von d), wenn es ein $b \in \mathbb{Z}$ mit $a = bd$ gibt. Um zu zeigen, daß a von d geteilt wird, schreibt man $d|a$.

Beispiel.

$2|8$, *wegen* $8 = 4 \cdot 2$

$5|-15$, *wegen* $-15 = -3 \cdot 5$

6 *dividiert* 17 *nicht, da es keine ganze Zahl* b *mit* $17 = b \cdot 6$ *gibt*

Satz 1. Die folgenden Beziehungen sind gültig:

1. $d|a$ und $a|b$ impliziert $d|b$ (Transitivität)
2. $d|1$, dann ist $d = \pm 1$
3. $d|a$ und $a|d$, dann ist $d = \pm a$
4. $d|a$ und $d|b$, dann $d|(ax + by)$ für beliebige ganze Zahlen x, y

Beweis. Stellvertretend soll 4 bewiesen werden. Die Annahme, daß $d|a$ und $d|b$ impliziert, daß es ganze Zahlen m, n gibt, mit $a = md$ und $b = nd$. Daraus folgt $ax + by = (md)x + (nd)y = (mx + ny)d$ und damit $d|(ax + by)$. □

Das Konzept des Divisors führt zur Definition der Primzahl.

Definition 2. Eine ganze positive Zahl p wird Primzahl (prim) genannt, wenn ihre einzigen beiden Divisoren 1 und p sind. Da die 1 ausgeschlossen wird, fangen die Primzahlen mit $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, \ldots$ an. Hat eine Zahl drei und mehr Divisoren, so wird sie zusammengesetzt genannt. Jede ganze Zahl größer als 1 ist entweder prim oder zusammengesetzt, jedoch niemals beides. Die 1 ist weder prim noch zusammengesetzt.

Der größte gemeinsame Teiler d zweier positiver ganzer Zahlen a, b wird wie folgt definiert:

Definition 3. Sind a, b ganze Zahlen, so wird für ihren größten gemeinsamen Teiler (*greatest common divisor*) $\gcd(a, b)$ geschrieben. Die ganze positive Zahl $d = \gcd(a, b)$ ist größter gemeinsamer Teiler von a und b, wenn

1. $d|a$ und $d|b$ und
2. jeder weitere Divisor c von a und b auch d teilt.

Da d eine ganze positive Zahl sein soll, gilt $d = \gcd(a, b) = \gcd(a, -b) = \gcd(-a, b) = \gcd(-a, -b)$.

Beispiel. $\gcd(12, 8) = 4$ *und* $\gcd(12, 9) = 3$.

Es gibt eine weitere Definition für den größten gemeinsamen Teiler. Sie beruht auf der Linearkombination zweier ganzer Zahlen.

Satz 2. Sind a, b ganze positive Zahlen, dann ist ihr größter gemeinsamer Teiler $\gcd(a, b)$ die kleinste ganze positive Zahl, die als Linearkombination $ax + by$, mit $x, y \in \mathbb{Z}$, ausgedrückt werden kann.

Beweis. Die Menge $M = \{ax + by \mid x, y \in \mathbb{Z},\ ax + by > 0\}$ hat zumindest das Element a^2+b^2 und ist somit nicht leer. M ist folglich eine nicht leere Untermenge von $\mathbb{N}$ und hat somit ein kleinstes Element $d = ax_1 + by_1$, mit $x_1 \in \mathbb{Z}$ und $y_1 \in \mathbb{Z}$ (sie ist wohlgeordnet).

Jetzt ist nur noch zu zeigen, daß $d = \gcd(a, b)$ ist. Wenn $d|a$, so gilt allgemein $a = qd+r$, mit $q \in \mathbb{Z}$ und $0 \leq r < d$. Dabei ist q der Quotient und r der Divisionsrest (siehe nächster Abschnitt). Es ist also $r = a - qd = a - q(ax_1 + by_1) = a(1 - qx_1) + b(-qy_1)$. Damit ist $r \in M$. Da aber $r < d$ ist, ergibt sich ein Widerspruch zur Annahme, daß d das kleinste Element von M ist. Es muß also $r = 0$ und damit $a = qd$ sein, womit $d|a$ bewiesen wäre. Die gleiche Argumentation gilt für $d|b$.

Als nächstes ist zu zeigen, daß wenn $c|a$ und $c|b$ auch $c|d$. Angenommen, $c|a$ und $c|b$, dann gibt es ganze Zahlen m und n mit $a = cm$ und $b = cn$. Es folgt $d = ax_1 + by_1 = cmx_1 + cny_1 = (mx_1 + ny_1)c$, das heißt, $c|d$. □

Beispiel.

$\gcd(3, 5) = 1$. *Daher gilt* $1 = 3 \cdot 2 + 5 \cdot (-1)$.

$\gcd(10, 35) = 5$. *Die Linearkombination lautet:* $5 = 10 \cdot (-3) + 35 \cdot 1$. *Satz 2 besagt, daß es keine kleinere positive ganze Zahl als* 5 *gibt (*1, 2, 3 *oder* 4*), die durch eine Linearkombination von* 10 *und* 35 *ausgedrückt werden kann.*

Definition 4. Zwei positive ganze Zahlen a, b werden teilerfremd (relativ prim) genannt, wenn für sie $\gcd(a, b) = 1$ gilt.

Aus der Definition folgt unmittelbar

Satz 3. Wenn a und b ganze Zahlen ungleich Null sind und p eine Primzahl mit $p|ab$, so folgt $p|a$ oder $p|b$.

Beweis. Wenn $p|a$, dann ist die Aussage bewiesen. Wenn p nicht a dividiert, so sind beide teilerfremd und es gilt $\gcd(a,p) = 1$, bzw. $1 = ax + py$ mit $x, y \in \mathbb{Z}$. Multipliziert man nun die Gleichung mit b, so folgt $b = (ab)x + p(by)$. Da aber $p|ab$ und $p|p$ muß $p|b$. □

Wenn also p eine Primzahl ist und das Produkt zweier ganzer Zahlen ungleich Null dividiert, so muß es wenigstens eine der beiden Zahlen teilen. Dies läßt sich auf das Produkt von mehr als zwei ganzen Zahlen erweitern.

Satz 4. Wenn $a_1, a_2, \ldots, a_n$ ganze Zahlen ungleich Null sind und p eine Primzahl mit $p|a_1a_2 \cdots a_n$, so muß $p|a_i$ für irgendein i, $1 \leq i \leq n$.

Beweis. Der Beweis erfolgt mit vollständiger Induktion. $S(n)$ sei folgende Aussage: Wenn $p|a_1a_2 \cdots a_n$, dann gibt es wenigstens ein i, $1 \leq i \leq n$, für das $p|a_i$ gilt. Für $S(1)$ heißt das, wenn $p|a_1$ dann $p|a_1$, was sicher stimmt.

Angenommen, $S(k)$ ist wahr. Jetzt muß noch bewiesen werden, daß $S(k+1)$ (also der Schritt von k nach $k+1$) ebenfalls wahr ist. Das heißt, es ist zu beweisen, daß wenn die Annahme wenn $p|a_1a_2 \cdots a_k$ dann $p|a_i$ für $1 \leq i \leq k$, wahr ist, daß dann auch die Aussage wenn $p|a_1a_2 \cdots a_ka_{k+1}$ dann $p|a_i$ für $1 \leq i \leq k+1$, stimmt.

Bei $p|(a_1a_2 \cdots a_k)(a_{k+1})$ handelt es sich um ein Produkt von zwei ganzen Zahlen ungleich Null. Es gilt also Satz 3 und damit entweder $p|(a_1a_2 \cdots a_k)$ oder $p|a_{k+1}$. Da $S(k)$ als wahr angenommen wird, folgt aus $p|(a_1a_2 \cdots a_k)$ unmittelbar $p|a_i$ für $1 \leq i \leq k$. Wenn $p|a_{k+1}$ dann $p|a_i$ für $1 \leq i \leq k+1$. □

Obwohl der Beweis ziemlich großen Aufwand für eine relativ klare Aussage bedeutet, ist er wegen der angewendeten Methode der vollständigen Induktion interessant. Sie basiert auf der Weitergabe von Wahrheiten. Es wird gezeigt, daß Aussage $S(1)$ wahr ist, was meistens problemlos ist. Dann wird angenommen, daß Aussage $S(k)$ wahr ist und gezeigt, daß die Wahrheit an $S(k+1)$ „weitergegeben“ wird. Zusammen mit der Tatsache, daß die Wahrheitskette mit einer wahren Aussage $S(1)$ beginnt folgt, daß auch $S(n)$ wahr sein muß.

Dieser Abschnitt wird mit dem Fundamentaltheorem der Arithmetik beschlossen.

Satz 5. Jede ganze Zahl $a > 1$ kann eindeutig (bis auf die Reihenfolge der Faktoren) durch ein Produkt von einer oder mehreren Primzahlen dargestellt werden.

Beweis. Es sei $M = \{x \in \mathbb{N} \mid x > 1$ und x kann nicht als Produkt einer oder mehrerer Primzahlen ausgedrückt werden$\}$. Es gilt zu beweisen, daß M leer ist.

Angenommen, M ist nicht leer. Dann muß es ein kleinstes Element c geben (die Menge ist wohlgeordnet). c kann keine Primzahl sein, da sie sonst nicht in M enthalten wäre. Also gibt es ganze Zahlen c_1, c_2, $1 < c_1 < c$ und $1 < c_2 < c$, mit $c = c_1c_2$. c ist also eine zusammengesetzte Zahl. c_1 und c_2 sind auch nicht in M enthalten, da sie kleiner als c sind und c ja das kleinste Element von M ist. Da sie nicht zu M gehören, können sie durch Primzahlen dargestellt werden. Dies gilt dann auch für c, da es ja ein Produkt von c_1 und c_2 ist.

Jetzt muß noch gezeigt werden, daß die Darstellung eindeutig ist. Angenommen, $c = p_1 p_2 \cdots p_r = q_1 q_2 \cdots q_t$ und alle p_i und q_i sind Primzahlen. Nach dem Kürzen aller gleicher Faktoren bleibt $c = p_1 p_2 \cdots p_s = q_1 q_2 \cdots q_m$ mit $1 \leq s \leq r$ und $1 \leq m \leq t$.

Es gilt $p_1 | p_1 p_2 \cdots p_s$ und damit $p_1 | q_1 q_2 \cdots q_m$. Laut Satz 4 muß daher $p_1 | q_\mu$, für irgendein μ, $1 \leq \mu \leq m$. Dies ist aber ein Widerspruch zu der Annahme, daß alle gemeinsamen Faktoren weggekürzt wurden. Die Darstellung ist folglich eindeutig und $M = \emptyset$. □

Beispiel. *Die Primfaktorisierung von* 280 *ist* $280 = 2 \cdot 2 \cdot 2 \cdot 5 \cdot 7 = 2^3 \cdot 5 \cdot 7$.

Die Primfaktorisierung erlaubt auch den größten gemeinsamen Teiler zweier ganzer Zahlen größer 1 *zu bestimmen. Angenommen, es ist* $\gcd(10, 36)$ *gesucht.* $10 = 2 \cdot 5$ *und* $36 = 2^2 \cdot 3^2$. *Die einzigen Teiler von* 10 *sind also* 2, 5 *und* $2 \cdot 5$, *von* 36 *sind es* 2, 2^2, 3, 3^2, $2 \cdot 3$, $2^2 \cdot 3$, $2 \cdot 3^2$ *und* $2^2 \cdot 3^2$. *Damit folgt für den* $\gcd(10, 36) = 2$.

3.4.2 Modulare Arithmetik

Im Mittelpunkt der modularen Arithmetik steht die Restklassenalgebra. Es werden also Divisionsreste von ganzen Zahlen $x \in \mathbb{Z}$ betrachtet. Dieser Rest kann wie folgt ausgedrückt werden:

$$x = y \underbrace{\lfloor x/y \rfloor}_{\text{Quotient}} + \underbrace{x \bmod y}_{\text{Rest}} \tag{3.1}$$

Bei „mod“ handelt es sich in diesem Fall um eine zweistellige (binäre) Operation, ebenso wie es die Addition oder Multiplikation sind. Diese gilt sowohl für positive als auch negative ganze Zahlen.

$$x \bmod y = x - y\lfloor x/y \rfloor, \quad \text{für } y \neq 0 \tag{3.2}$$

Beispiel.

$$\begin{aligned}
5 \bmod 3 &= 5 - 3\lfloor 5/3 \rfloor = 5 - 3\lfloor 1.6 \rfloor = 2 \quad \textit{(siehe Abbildung 3.1)} \\
5 \bmod -3 &= 5 - (-3)\lfloor 5/(-3) \rfloor = 5 - (-3)\lfloor -1.6 \rfloor = 5 - (-3)(-2) = -1 \\
-5 \bmod 3 &= -5 - 3\lfloor -5/3 \rfloor = -5 - 3\lfloor -1.6 \rfloor = -5 - 3(-2) = 1 \\
-5 \bmod -3 &= -5 - (-3)\lfloor -5/(-3) \rfloor = -5 - (-3)\lfloor 1.6 \rfloor = -5 - (-3)(1) = -2
\end{aligned}$$

Wird „mod“ in Gleichungen verwendet, so empfiehlt sich eine etwas andere Schreibweise.

$$x \equiv y \pmod{n} \qquad \Longleftrightarrow \qquad x \bmod n = y \bmod n \tag{3.3}$$

Diese Schreibweise stammt von Gauß und wird „x kongruent y modulo n“ gelesen. Gauß schrieb dazu:

> *Numerorum congruentiam hoc signo,* $\equiv$, *in posterum denotabimus, modulum ubi opus erit in clausulis adiungentes,* $-16 \equiv 9 \pmod 5$, $-7 \equiv 15 \pmod{11}$.

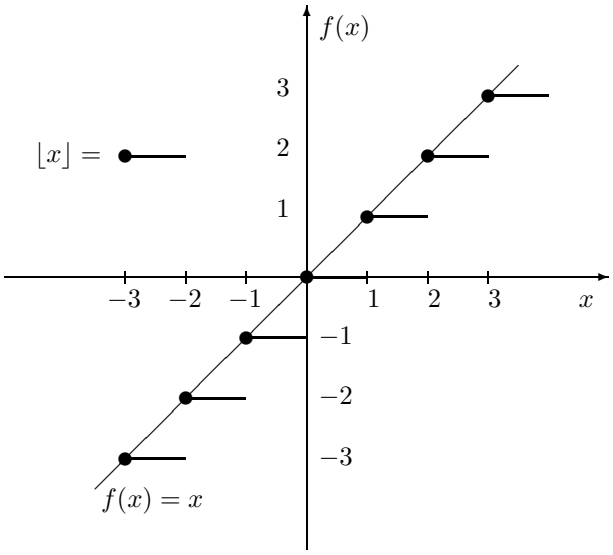


Abbildung 3.1: Rundungen auf ganze Zahlen

Eingedeutscht heißt „x kongruent y modulo n“ in etwa „x übereinstimmend (äquivalent, entsprechend) y mit dem Maß (gemessen an) n“.

Gemessen an heißt, daß die Rechnung in der Menge $\mathbb{Z}_n = \{0, 1, \ldots, (n-1)\}$ der Reste modulo n erfolgt.

Beispiel.

$$\begin{aligned} -16 \bmod 5 &= -16 - 5\lfloor -16/5 \rfloor = -16 - 5\lfloor -3.2 \rfloor = -16 - 5(-4) = -16 + 20 = 4 \\ &= 9 \bmod 5 = 9 - 5\lfloor 9/5 \rfloor = 9 - 5(1) = 4 \end{aligned}$$

-16 *hat den gleichen Rest bei Division mit* 5 *wie* 9*, daher gilt*

$$9 \equiv -16 \pmod 5$$

Bei gleichen Resten heben sich diese bei der Subtraktion von x und y weg und für $x - y$ ergibt sich ein Vielfaches von n, $x - y$ ist also durch n teilbar. Daraus folgt

$$x \equiv y \pmod n \qquad \Longleftrightarrow \qquad x - y \quad \text{ist ein Vielfaches von } n \text{ (teilbar durch } n\text{).} \tag{3.4}$$

Beispiel. *Wegen* $9 - (-16) = 25 = 5 \cdot 5$ *gilt* $9 \equiv -16 \pmod 5$*, da beide Seiten der Gleichung durch* 5 *teilbar sind (bzw.* $9 - (-16)$ *ein Vielfaches von* 5 *ist).*

Wenn der Dividend x ein Vielfaches des Divisors n enthält, gilt

$$x = kn + m, \quad x, k \in \mathbb{Z},\ n \in \mathbb{N},\ 0 \le m \le (n-1), \quad (kn + m) \bmod n = m \bmod n \tag{3.5}$$

Beispiel.

$$15 \bmod 11 = (11 \cdot 1 + 4) \bmod 11 = 4 = -7 \bmod 11 = (11 \cdot 1 - 7) \bmod 11 = 4 \bmod 11 = 4$$

Der modulo n Operator bildet folglich alle ganzen Zahlen $\mathbb{Z}$ nach $\mathbb{Z}_n$ ab. Innerhalb dieser Menge gelten die Regeln der allgemeinen Arithmetik für die Addition, Subtraktion und Multiplikation.

Wenn $m \in \mathbb{Z}_n$ und $\gcd(m,n) = 1$, dann hat m eine eindeutige multiplikative Inverse in $\mathbb{Z}_n$, für die m^{-1} geschrieben wird.

$$\gcd(m,n) = 1, \text{ dann gibt es für } m \in \mathbb{Z}_n \text{ ein } b \text{ mit } mb \equiv 1 \pmod{n} \text{ und } b = m^{-1}. \tag{3.6}$$

Beweis. Da $\gcd(m,n) = 1$ ist, kann 1 als Linearkombination von m und n ausgedrückt werden. Das heißt, daß es ganze Zahlen r und t gibt, für die $1 = rm+nt$ bzw. $1-rm = nt$ und somit $1 \equiv rm \pmod{n}$. Dies gilt dann und nur dann, wenn m teilerfremd zu n ist. Ist $n = p$ eine Primzahl, so hat jedes Element außer 0 aus $\mathbb{Z}_n$ eine multiplikative Inverse. □

Beispiel. *Für die multiplikativen und additiven Inversen modulo 7 folgt:*

m	$-m$	m^{-1}
0	0	–
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

Für $m = 2$ *gilt z.B.* $2 \cdot 4 \bmod 7 = 1 \bmod 7$ *und damit* $2^{-1} = 4$. *Außerdem gilt* $(2 + 5) \bmod 7 = 0 \bmod 7$. *Die additive Inverse von* 2 *modulo* 7 *ist also* $-2 = 5$.

Für jedes $m \in \mathbb{Z}_n$ *gibt es ein* b *mit* $m + b \equiv 0 \pmod{n}$, *geschrieben* $b = -m$. (3.7)

Beispiel. *In* $\mathbb{Z}_{26}$ *gibt es dagegen nur für folgende Zahlen multiplikative Inverse:* $1^{-1} = 1$, $3^{-1} = 9$, $5^{-1} = 21$, $7^{-1} = 15$, $11^{-1} = 19$, $17^{-1} = 23$ *und* $25^{-1} = 25$. *Es gilt z.B.* $7 \cdot 15 = 105 \equiv 1 \pmod{26}$ *und daher* $7^{-1} = 15$.

Bevor ich nun einige Rechenregeln der modularen Arithmetik anführe, möchte ich zuerst etwas zur Nomenklatur sagen. Diese ist leider nicht in allen Büchern einheitlich, oft wird sogar in ein und demselben Buch die Schreibweise nach belieben geändert. In [20] wird z.B. überall „=“ verwendet. Gleichung (3.3) wird so zu $x = y \pmod{n}$ und für den unbedarften Leser fängt das Chaos an. In [9] wird einmal „=“ und ein anderes Mal „≡“ verwendet, vermutlich je nachdem, welche Quelle der Autor verwendet hat. Ein Versuch, ihn darauf hinzuweisen, stoß auf Unverständnis. Eine gewisse Vorsicht ist auch bei der Verwendung von „≡“ geboten, da z.B. für die Äquivalenzrelation mitunter das gleiche Symbol (neben „$\Longleftrightarrow$“) verwendet wird.

Und nun zu den Rechenregeln.

Satz 6. Es gilt $n \in \mathbb{N}$ und $w, x, y, z \in \mathbb{Z}$.

1. $x \equiv x \pmod{n}$

2. $x \equiv y \pmod{n}$ impliziert $y \equiv x \pmod{n}$
3. $x \equiv y \pmod{n}$ und $y \equiv z \pmod{n}$ impliziert $x \equiv z \pmod{n}$ (Transitivität)
4. $x \equiv y \pmod{n}$ und $w \equiv z \pmod{n}$ impliziert $x + w \equiv y + z \pmod{n}$ und $xw \equiv yz \pmod{n}$
5. $x \equiv y \pmod{n}$ impliziert $xz \equiv yz \pmod{n}$
6. $x \equiv y \pmod{n}$ impliziert $x^k \equiv y^k \pmod{n}$ für $k \in \mathbb{N}$
7. $[(x \bmod n) \pm (y \bmod n)] \bmod n = (x \pm y) \bmod n$
8. $[(x \bmod n)(y \bmod n)] \bmod n = (xy) \bmod n$

Beweis. Exemplarisch sollen die Regeln 3, 4, 6 und 8 bewiesen werden.

Zuerst Regel 3. Wegen (3.4) gibt es zwei ganze Zahlen r, s mit $x-y = rn$ und $y-z = sn$. Addiert man die beiden Gleichungen, so erhält man $x - z = (r + s)n$, woraus $x \equiv z \pmod{n}$ folgt.

Für Regel 4 gilt analog $x - y = rn$ und $w - z = sn$ mit $r, s \in \mathbb{Z}$. Durch Addition der beiden Gleichungen erhält man $(x + w) - (y + z) = (r + s)n$ und damit $x + w \equiv y + z \pmod{n}$.

Der zweite Teil von Regel 4 besagt, daß $xw \equiv yz \pmod{n}$. Mit (3.4) und $r \in \mathbb{Z}$ folgt $xw - yz = rn$. Durch Erweiterung der linken Seite der Gleichung mit $yw - yw = 0$ erhält man $xw - yw + yw - yz = rn = (x - y)w + y(w - z)$. Da n, wie oben gezeigt, $x - y$ und $w - z$ teilt, ist der zweite Teil der Regel bewiesen.

Der Beweis des zweiten Teils von Regel 4 kann zum Beweis von Regel 6 verwendet werden. Dazu ist $w = x$ und $z = y$ zu setzen. Dann gilt mit $r \in \mathbb{Z}$ und der Erweiterung von oben $x^2 - y^2 = (x - y)x + y(x - y) = rn$ und damit $x^2 \equiv y^2 \pmod{n}$. Für $w = x^2$ und $z = y^2$ folgt analog $x^3 - y^3 = (x - y)x^2 + y(x^2 - y^2)$. Setzt man nun das Ergebnis aus dem vorhergehenden Schritt, $x^2 - y^2 = (x - y)x + y(x - y)$ in die Gleichung ein, so erhält man $x^3 - y^3 = (x - y)x^2 + y(x - y)x + y^2(x - y)$ und damit $x^3 \equiv y^3 \pmod{n}$. Fährt man mit diesem Verfahren fort, so erhält man $x^k \equiv y^k \pmod{n}$.

Für Regel 8 wird zuerst $x \bmod n = r_x$ und $y \bmod n = r_y$ gesetzt. Mit (3.1) folgt dann $x = jn + r_x$ und $y = kn + r_y$ für beliebige $j, k \in \mathbb{Z}$. Daraus folgt

$$\begin{aligned}[(x \bmod n)(y \bmod n)] \bmod n &= [(x - jn)(y - kn)] \bmod n \\ &= [xy - xkn - jny + jknn] \bmod n, \quad \text{und wegen (3.5)} \\ &= (xy) \bmod n\end{aligned}$$

□

3.4.3 Mehr Zahlentheorie

Eine weitere wichtige Größe der Zahlentheorie ist die eulersche Funktion ϕ (im Englischen auch *totient function* genannt).

Definition 5. Die Funktion $\phi(n)$ ist die Anzahl ganzer positiver Zahlen, die kleiner als n und zu n teilerfremd sind.

Beispiel. $\phi(7) = 6$, *da es sechs ganze positive Zahlen (*$M = \{1, 2, 3, 4, 5, 6\}$*) gibt, für die* $\gcd(x, 7) = 1$, $x \in M$, *ist.*

$\phi(6) = 2$, *da es nur zwei ganze positive Zahlen gibt, die zu* 6 *teilerfremd sind, nämlich* 1 *und* 5.

$\phi(16) = 8$, *da* $1, 3, 5, 7, 9, 11, 13$ *und* 15 *die einzigen ganzen positiven Zahlen sind, die kleiner als* 16 *und zu* 16 *teilerfremd sind.*

Für $\phi(n)$ gibt es einige Rechenregeln, die im folgenden hergeleitet und bewiesen werden.

Satz 7. Ist p eine Primzahl, so gilt $\phi(p) = p - 1$.

Beweis. Die einzigen Teiler von p sind p und 1. Daher sind alle ganzen positiven Zahlen kleiner p (nämlich, $1, 2, \ldots, p-1$) zu p teilerfremd. □

Satz 8. Sind p und q unterschiedliche Primzahlen, so gilt $\phi(pq) = (p-1)(q-1) = \phi(p)\phi(q)$.

Beweis. Es gibt $pq-1$ ganze Zahlen kleiner pq. Davon sind die Zahlen $p, 2p, 3p, \ldots, (q-1)p$ durch p teilbar und $q, 2q, 3q, \ldots, (p-1)q$ durch q. Es bleiben also $pq - 1 - (q-1) - (p-1) = pq - p - q + 1 = (p-1)(q-1)$ Zahlen übrig, die weder durch p noch durch q geteilt werden können und daher mit pq teilerfremd sind. □

Beispiel. $\phi(3 \cdot 5) = \phi(15) = \phi(3)\phi(5) = 2 \cdot 4 = 8$, *nämlich* $1, 2, 4, 7, 8, 11, 13, 14$.

Satz 9. Falls p eine Primzahl und k eine ganze positive Zahl ist, so gilt $\gcd(n, p^k) = 1$ dann und nur dann, wenn p nicht n teilt.

Beweis. Sollte $p|n$, so wäre $\gcd(n, p^k) \neq 1$, da p auch p^k teilt. Umgekehrt, wenn $\gcd(n, p^k) \neq 1$, dann sind n und p^k nicht teilerfremd und sie müssen einen gemeinsamen Faktor größer 1 haben. Da aber die einzigen Faktoren von p^k Potenzen von p sind, gilt $p|n$. □

Satz 10. Ist p eine Primzahl und k eine positive ganze Zahl, so gilt $\phi(p^k) = p^k - p^{k-1}$.

Beweis. Von 1 bis p^k gibt es p^{k-1} ganze Zahlen $(p, 2p, 3p, \ldots, p^{k-1}p^k)$, die durch p teilbar sind. Dazu gehört auch p^k selbst. Es bleiben also $p^k - p^{k-1}$ Zahlen übrig, die positiv und kleiner als p^k sind. □

Beispiel. *Für* $p = 2$ *und* $k = 3$ *folgt* $p^k = 2^3 = 8$ *mit* $\phi(8) = 4$. *Das gleiche Ergebnis erhält man mit* $p^k - p^{k-1} = 2^3 - 2^2 = 4$.

Satz 11. Für ganze Zahlen a, b und c gilt $\gcd(a, bc) = 1$ dann und nur dann, wenn auch $\gcd(a, b) = 1$ und $\gcd(a, c) = 1$.

Beweis. Angenommen, $\gcd(a, bc) = 1$. Es ist zu zeigen, daß $\gcd(a, b) = 1$. Angenommen, $\gcd(a, b) = d$. Dann gilt $d|a$ und $d|b$. Da aber $\gcd(a, bc) = 1$ ist, muß $d = 1$, nämlich der größte gemeinsame Teiler von a und bc, sein. Die gleiche Argumentation gilt für $\gcd(a, c) = 1$.

Umgekehrt sei angenommen, daß $\gcd(a,b) = 1$ und $\gcd(a,c) = 1$. Es ist zu zeigen, daß $\gcd(a,bc) = 1$. Angenommen, $\gcd(a,bc) = d > 1$. Dann muß d eine Primzahl p enthalten, die $p|a$ und $p|bc$. Also gilt $p|b$ oder $p|c$. Wenn aber $p|b$ und $p|a$ so muß wegen $\gcd(a,b) = 1$ auch $p|1$ und kann daher keine Primzahl sein. Wenn andererseits $p|c$ und $p|a$, so muß wegen $\gcd(a,c) = 1$ wiederum $p|1$, was ausschließt, daß p eine Primzahl ist. Die Annahme $\gcd(a,bc) > 1$ führt also zu einem Widerspruch und es ist daher $\gcd(a,bc) = 1$. □

Beispiel. *Angenommen* $a = 5$, $b = 6$ *und* $c = 7$. *Dann ist* $\gcd(5,6) = 1 = \gcd(5,7)$. *Es muß also auch* $\gcd(5, 6 \cdot 7) = \gcd(5,42) = 1$ *sein.*

Ist andererseits $a = 5$, $b = 6$ *und* $c = 15$, *so folgt* $\gcd(5,6) = 1$ *aber* $\gcd(5,15) = 5$ *und damit* $\gcd(5,90) = 5 \neq 1$.

Jetzt wird es noch spannender.

Satz 12. Wenn m und n teilerfremde positive ganze Zahlen sind, so gilt $\phi(mn) = \phi(m)\phi(n)$.

Beweis. Die Zahlen von 1 bis mn sind $1, 2, \ldots, r, \ldots, mn$. Ordnet man diese Zahlen in einer $n \times m$-Matrix mit n Zeilen und m Spalten an, so erhält man

$$\begin{array}{ccccccc} 1 & 2 & \ldots & r & \ldots & m \\ m+1 & m+2 & \ldots & m+r & \ldots & 2m \\ 2m+1 & 2m+2 & \ldots & 2m+r & \ldots & 3m \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (n-1)m+1 & (n-1)m+2 & \ldots & (n-1)m+r & \ldots & nm \end{array}$$

$\phi(mn)$ ist die Anzahl der Elemente a_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$, die teilerfremd zu mn sind, für die also $\gcd(a_{ij}, mn) = 1$ gilt. Laut Satz 11 muß dann aber auch $\gcd(a_{ij}, m) = 1$ und $\gcd(a_{ij}, n) = 1$ erfüllt sein.

Wenn d die kleinste ganze Zahl ist, die als Linearkombination von $qm + r$ und m ausgedrückt werden kann, $d = a(qm + r) + bm$, $a, b \in \mathbb{Z}$, so ist d auch die kleinste ganze Zahl, die eine Linearkombination von m und r, $d = (aq + b)m + ar$, ist. Es gilt also $\gcd(qm+r, m) = \gcd(r, m)$. Das heißt aber, daß die Zahlen in der r-ten Spalte nur dann teilerfremd zu m sind, wenn r selbst es ist. Es gibt also $\phi(m)$ Spalten, die teilerfremd zu m sind und jede Zahl in diesen Spalten ist es ebenfalls.

Es ist als noch zu zeigen, daß in jeder der $\phi(m)$ Spalten $\phi(n)$ Elemente teilerfremd zu n sind. Nur dann sind $\phi(m)\phi(n)$ Elemente teilerfremd zu beiden, m und n.

Angenommen, $\gcd(r, m) = 1$. Es sollen dann die Elemente der r-ten Spalte, $r, m + r, 2m + r, \ldots, (n-1)m + r$ auf Teilerfremdheit zu n untersucht werden. Die Spalte hat n Elemente, wobei keine zwei kongruent modulo n zueinander sind. Falls sie es wären, müßte nämlich $km + r \equiv jm + r \pmod{n}$ bzw. $km \equiv jm \pmod{n}$, $0 \leq j < k < n$, sein. Das heißt, es gäbe eine ganze Zahl t, für die $km - jm = tn$ bzw. $m(k-j) = tn$. Da $n|tn$ muß auch $n|m(k-j)$. Es ist aber $\gcd(m,n) = 1$ und so kann nur $n|(k-j)$ woraus $k - j = un$, $u \in \mathbb{Z}$, folgt. Dies heißt aber, daß $k \equiv j \pmod{n}$, was aber ein Widerspruch ist, da aus $0 \leq j < k < n$ auch $k - j < n$ folgt und n daher nicht $k - j$ teilen kann.

Die Zahlen a_{ir}, $1 \leq i \leq n$, in der r-ten Spalte sind folglich in irgendeiner Reihenfolge kongruent modulo n zu den Elementen von $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$. Für $s \in \mathbb{Z}_n$ und $s \equiv a_{ir} \pmod{n}$ gilt $\gcd(a_{ir}, n) = 1$ dann und nur dann, wenn auch $\gcd(s, n) = 1$ ist. Denn es folgt aus $\gcd(s, n) = 1$, daß es zwei ganze Zahlen a, b gibt, für die $as + bn = 1$. Für $v \in \mathbb{Z}$ folgt aus $s \equiv a_{ir} \pmod{n}$ aber $s - a_{ir} = vn$ und damit $a(a_{ir} + vn) + bn = aa_{ir} + (av + b)n = 1$, woraus wiederum $\gcd(a_{ir}, n) = 1$ folgt.

Die r-te Spalte enthält also soviele ganze Zahlen, die teilerfremd zu n sind, wie sie die Menge $\mathbb{Z}_n$ enthält, nämlich $\phi(n)$. Daher ist die Gesamtzahl ganzer Zahlen, die teilerfremd zu m und n sind, $\phi(m)\phi(n)$. □

Beispiel.

$$\begin{aligned}
&m = 3,\ n = 4, && \phi(3)\phi(4) = 2 \cdot 2 = 4 = \phi(12) = \phi(3 \cdot 4)\\
&m = 7,\ n = 2, && \phi(7)\phi(2) = 6 \cdot 1 = 6 = \phi(14) = \phi(7 \cdot 2)\\
&m = 4,\ n = 2, && \phi(4)\phi(2) = 2 \cdot 1 = 2 \neq \phi(4 \cdot 2) = \phi(8) = 4,\ da\ \gcd(4, 2) = 2 \neq 1
\end{aligned}$$

Satz 12 kann auf ein Produkt von Primzahlen erweitert werden.

Satz 13. Wenn es für $n > 1$ die Primzahlzerlegung $n = p_1^{k_1} \cdot p_2^{k_2} \ldots p_r^{k_r}$ gibt, so ist $\phi(n) = (p_1^{k_1} - p_1^{k_1-1})(p_2^{k_2} - p_2^{k_2-1}) \cdots (p_r^{k_r} - p_r^{k_r-1})$.

Beweis. Der Beweis erfolgt mit vollständiger Induktion. Der Induktionsbeginn für $r = 1$ ist wegen Satz 10 erfüllt.

Angenommen, die Beziehung gilt für $r = m$. Das heißt, wenn $n = p_1^{k_1} \cdot p_2^{k_2} \ldots p_m^{k_m}$ folgt $\phi(n) = (p_1^{k_1} - p_1^{k_1-1})(p_2^{k_2} - p_2^{k_2-1}) \cdots (p_m^{k_m} - p_m^{k_m-1})$.

Für $r = m + 1$ ist dann $n = (p_1^{k_1} \cdot p_2^{k_2} \ldots p_m^{k_m}) p_{m+1}^{k_{m+1}}$ und mit Satz 12 folgt $\phi(n) = \phi(p_1^{k_1} \cdot p_2^{k_2} \ldots p_m^{k_m})\phi(p_{m+1}^{k_{m+1}})$. Satz 10 ergibt zusammen mit der Induktionsvoraussetzung $\phi(n) = (p_1^{k_1} - p_1^{k_1-1})(p_2^{k_2} - p_2^{k_2-1}) \cdots (p_{m+1}^{k_{m+1}} - p_{m+1}^{k_{m+1}-1})$. □

Mit Hilfe von Satz 13 kann $\phi(n)$ auch für große n relativ einfach bestimmt werden.

Beispiel.

$$\begin{aligned}
\phi(360) &= \phi(2^3 \cdot 3^2 \cdot 5) = (2^3 - 2^2)(3^2 - 3^1)(5^1 - 5^0) = 4 \cdot 6 \cdot 4 = 96\\
\phi(1575) &= \phi(3^2 \cdot 5^2 \cdot 7) = (3^2 - 3^1)(5^2 - 5^1)(7^1 - 7^0) = 6 \cdot 20 \cdot 6 = 720
\end{aligned}$$

Satz 14. Für $n > 2$ ist $\phi(n)$ eine gerade Zahl.

Beweis. Wenn n eine Zweierpotenz ist, $n = 2^k$, $k > 1$, dann folgt aus Satz 10 $\phi(n) = \phi(2^k) = 2^k - 2^{k-1} = 2^{k-1}(2 - 1) = 2^{k-1}$, was eine gerade Zahl ist.

Ist n keine Zweierpotenz, so muß n durch eine ungerade Primzahl p teilbar sein, $n = p^k m$, $k \geq 1$. Dann gilt $\gcd(p^k, m) = 1$ und damit $\phi(n) = \phi(p^k m) = (p^k - p^{k-1})\phi(m) = p^{k-1}(p - 1)\phi(m)$. Dies ist aber eine gerade Zahl, da $2|(p - 1)$. □

Satz 15. Es sei $n > 1$ und $\gcd(a, n) = 1$. Wenn $a_1, a_2, \ldots, a_{\phi(n)}$ die positiven ganzen Zahlen kleiner n und teilerfremd zu n sind, so sind die Zahlen $aa_1, aa_2, \ldots, aa_{\phi(n)}$ kongruent modulo n zu $a_1, a_2, \ldots, a_{\phi(n)}$ in irgendeiner Reihenfolge.

Beweis. Zuerst ist festzustellen, daß keine zwei Zahlen aus $\{aa_1, aa_2, \ldots, aa_{\phi(n)}\}$ kongruent modulo n zueinander sind. Wäre dem nicht so, so würde $aa_i \equiv aa_j \pmod{n}$ für $i \neq j$ sein und damit $aa_i - aa_j = a(a_i - a_j) = kn$ für $k \in \mathbb{Z}$. Da aber $n|kn$, muß auch $n|a(a_i - a_j)$. Wegen $\gcd(a, n) = 1$ kann aber nur $n|(a_i - a_j)$, woraus $a_i - a_j = tn$, $t \in \mathbb{Z}$ und damit $a_i \equiv a_j \pmod{n}$ folgt. Dies ist aber ein Widerspruch, da $1 \leq a_j < a_i < n$ und damit $a_i - a_j < n$, woraus folgt, daß n nicht $a_i - a_j$ teilen kann. Eine ähnliche Schlußfolgerung führte schon in Satz 12 zum Ziel.

Da $\gcd(a, n) = 1 = \gcd(a_1, n) = \gcd(a_2, n) = \cdots = \gcd(a_{\phi(n)}, n)$ ist, garantiert Satz 11 ferner $\gcd(aa_1, n) = \gcd(aa_2, n) = \cdots = \gcd(aa_{\phi(n)}, n) = 1$.

Angenommen, es gilt $aa_i \equiv b \pmod{n}$ für irgendein $b \in \mathbb{Z}_n$ und damit $aa_i - b = kn$, $k \in \mathbb{Z}$. Damit ist $1 = \gcd(aa_i, n) = \gcd(b, n)$, weil für ganze Zahlen x, y die Linearkombination $1 = x(aa_i) + yn = x(b + kn) + yn = xb + (kx + y)n$ ist. Damit muß aber b eine der Zahlen $a_1, a_2, \ldots, a_{\phi(n)}$ sein, weil dies die einzigen Zahlen sind, die teilerfremd zu n sind. Daher ist $aa_i \equiv b \pmod{n}$ mit $b \in \{a_1, a_2, \ldots, a_{\phi(n)}\}$. □

Beispiel. *Es sei* $n = 8$ *und* $a = 5$ *und damit* $\gcd(5, 8) = 1$. *Alle positiven ganzen Zahlen kleiner* 8 *und teilerfremd zu* 8 *sind* $S = \{1, 3, 5, 7\}$. *Multipliziert man nun jedes Element von* S *mit* 5, *so erhält man* $R = \{5, 15, 25, 35\}$. *Satz 15 besagt, daß jedes Element aus* R *kongruent modulo* 8 *zu einem Element aus* S *sein muß. Dies trifft tatsächlich zu, da* $5 \equiv 5 \pmod{8}$, $15 \equiv 7 \pmod{8}$, $25 \equiv 1 \pmod{8}$ *und* $35 \equiv 3 \pmod{8}$.

Es ist geschafft. Nun zu dem Satz, der diesen ganzen Aufwand rechtfertigt, zum **Satz von Euler**.

Satz 16. Wenn n eine ganze positive Zahl mit $\gcd(a, n) = 1$ ist, dann gilt

$$a^{\phi(n)} \equiv 1 \pmod{n}. \tag{3.8}$$

Beweis. Der Satz gilt für $n = 1$, da mit $\phi(1) = 0$ unmittelbar $a^0 \equiv 1 \pmod{1}$ folgt. Es ist zu erwähnen, daß für $\phi(1)$ nicht überall Null gesetzt wird, wie dies z.B. in [9] der Fall ist. In [6] wird $\phi(1) = 1$ gesetzt und in [18] ist $\phi(n)$ nur für $n \geq 2$ definiert. Dies tut aber der Gültigkeit von (3.8) keinen Abbruch, da auch $a^1 \equiv 1 \pmod{1}$ für $k \in \mathbb{Z}$ wegen $a - 1 = k$ erfüllt ist, nämlich für $k = a - 1$.

Für $n > 1$ sind $a_1, a_2, \ldots, a_{\phi(n)}$ die positiven ganzen Zahlen kleiner n, die teilerfremd zu n sind.

$\gcd(a, n) = 1$ vorausgesetzt, besagt Satz 15, daß die Werte $aa_1, aa_2, \ldots, aa_{\phi(n)}$ in irgendeiner Reihenfolge kongruent modulo n zu $a_1, a_2, \ldots, a_{\phi(n)}$ sind. Das heißt,

$$\begin{aligned} aa_1 &\equiv a'_1 \pmod{n} \\ aa_2 &\equiv a'_2 \pmod{n} \\ &\vdots \\ aa_{\phi(n)} &\equiv a'_{\phi(n)} \pmod{n} \end{aligned}$$

Wegen Satz 6, Regel 4, können die Terme links und rechts der Äquivalenz miteinander multipliziert werden. Daraus folgt

$$aa_1aa_2\cdots aa_{\phi(n)} \equiv a'_1a'_2\cdots a'_{\phi(n)} \pmod{n} \quad \text{bzw.}$$
$$aa_1aa_2\cdots aa_{\phi(n)} \equiv a_1a_2\cdots a_{\phi(n)} \pmod{n}$$

und damit

$$a^{\phi(n)}(a_1a_2\cdots a_{\phi(n)}) \equiv a_1a_2\cdots a_{\phi(n)} \pmod{n}$$

Setzt man $x = a_1a_2\cdots a_{\phi(n)}$ so folgt $a^{\phi(n)}x \equiv x \pmod{n}$ und wegen Satz 11 auf Seite 183 $\gcd(x,n) = 1$.

Damit ist $a^{\phi(n)}x - x = kn$ bzw. $x(a^{\phi(n)}-1) = kn$ für $k \in \mathbb{Z}$. Da $n|kn$ muß auch $n|xa^{\phi(n)}$. Weil aber $\gcd(n,x) = 1$ kann nur $n|(a^{\phi(n)}-1)$ und damit $a^{\phi(n)} \equiv 1 \pmod{n}$. □

Beispiel. *Für $a = 5$, $n = 8$ ist $\gcd(5,8) = 1$ und $\phi(8) = 4$ bzw. $5^{\phi(8)} = 5^4 = 625$. Somit ist $625 \equiv 1 \pmod 8$ weil $625 - 1 = 624 = 8 \cdot 78$.*

Für $a = 2$, $n = 7$ ist $\gcd(2,7) = 1$ und $\phi(7) = 6$ bzw. $2^{\phi(7)} = 2^6 = 64$. Somit ist $64 \equiv 1 \pmod 7$ weil $64 - 1 = 63 = 9 \cdot 7$.

Ist p eine Primzahl, die a nicht teilt, $\gcd(a,p) = 1$, gilt der Satz von Euler und es folgt $a^{\phi(p)} \equiv 1 \pmod{p}$. Wegen Satz 7 gilt dann $a^{p-1} \equiv 1 \pmod{p}$. Da dies Ergebnis unabhängig von Euler 1640 von Fermat entdeckt wurde, wird es **Satz von Fermat** (manchmal auch der kleine Satz von Fermat) genannt.

Satz 17. Ist p eine Primzahl mit $\gcd(a,p) = 1$, so gilt

$$a^{p-1} \equiv 1 \pmod{p}. \tag{3.9}$$

Beispiel. *Ist $p = 5$ und $a = 2$, so folgt $a^{\phi(p)} = 2^4 = 16 \equiv 1 \pmod 5$.*

Ist n das Produkt zweier Primzahlen, so führt Satz 8 zu folgendem **Zusatz zum Satz von Euler**:

Satz 18. Ist a teilerfremd zu den beiden unterschiedlichen Primzahlen p und q, so gilt

$$a^{(p-1)(q-1)} \equiv 1 \pmod{pq}. \tag{3.10}$$

Beispiel. *Sei $p = 13$, $q = 113$ und $a = 939$. Da $a = 939 = 3 \cdot 313$ ist, ist es teilerfremd zu den beiden Primzahlen 13 und 113 und es gilt $939^{12\cdot 112} = 939^{1344} \equiv 1 \pmod{1469}$.*

3.4.4 Der RSA-Algorithmus

Alle Berechnungen des RSA-Algorithmus finden in der Menge $\mathbb{Z}_n$ statt, wobei $n = pq$ ist und p, q unterschiedliche große Primzahlen. Für die eulersche Funktion gilt dann $\phi(n) = (p-1)(q-1)$.

Satz 19. Es sei $m < n$ die zu verschlüsselnde Nachricht, $n = pq$, p, q unterschiedliche Primzahlen. Der öffentliche Schlüssel sei e, $1 < e < \phi(n)$ mit $\gcd(e, \phi(n)) = 1$, der private d, mit $ed \equiv 1 \pmod{\phi(n)}$ bzw. $d \equiv e^{-1} \pmod{\phi(n)}$. Wenn $c = m^e \bmod n$ und $m' = c^d \bmod n$ dann $m' = m$.

Beweis. Es ist $m' = c^d \bmod n = (m^e \bmod n)^d \bmod n$ und wegen Satz 6, Regel 8, $m' = m^{ed} \bmod n$. Wenn gezeigt werden kann, daß $m^{ed} \equiv m \pmod{n}$, dann ist $m' = m$.

Zuerst soll gezeigt werden, daß $m^{ed} \equiv m \pmod{p}$. Wegen $ed \equiv 1 \pmod{\phi(n)}$ gibt es eine ganze Zahl t mit $ed - 1 = t\phi(n)$ bzw. $ed = 1 + t\phi(n) = 1 + t(p-1)(q-1)$.

Im Fall, daß m und p nicht teilerfremd sind ($p|m$), ist $m \equiv 0 \pmod{p}$ und $m^{ed} \equiv 0 \pmod{p}$ und damit wegen Satz 6, Regel 3, $m^{ed} \equiv m \pmod{p}$.

Sind m und p teilerfremd ($\gcd(m, p) = 1$), gilt

$$\begin{aligned} m^{ed} \bmod p &= m^{1+t(p-1)(q-1)} \bmod p \\ &= m \cdot m^{t(p-1)(q-1)} \bmod p \\ &= m(m^{p-1})^{t(q-1)} \bmod p \\ &= m(m^{\phi(p)})^{t(q-1)} \bmod p \end{aligned}$$

und mit Satz 6, Regel 8,

$$= \Big\{(m \bmod p)\big[(m^{\phi(p)})^{t(q-1)} \bmod p\big]\Big\} \bmod p$$

sowie (3.8) und Satz 6, Regel 6 und 8,

$$\begin{aligned} &= \big[(m \bmod p)(1^{t(q-1)} \bmod p)\big] \bmod p \\ &= m \bmod p \end{aligned}$$

Die gleiche Schlußfolgerung ergibt $m^{ed} \equiv m \pmod{q}$.

Da aber $p|(m^{ed} - m)$ und $q|(m^{ed} - m)$ folgt, daß $pq|(m^{ed} - m)$, da p und q unterschiedliche Primzahlen sind. Es gibt also eine ganze Zahl r mit $m^{ed} - m = (pq)r = nr$ und damit ist $m \equiv m^{ed} \pmod{n}$. □

Beispiel. *Da Bob und Alice wohl zu denen gehören, die ihren Nachrichtenverkehr verschlüsseln, sollen sie auch in diesem Beispiel auftreten. Bob wählt* $p = 101$ *und* $q = 113$*, woraus* $n = pq = 11413$ *und* $\phi(n) = \phi(11413) = 100 \cdot 112 = 11200$ *folgt. Um* e *zu erhalten, können z.B. alle Zahlen ermittelt werden, die* $\phi(n)$ *teilen.* e *wird dann aus den Zahlen gewählt, die nicht zu den Divisoren von* $\phi(n)$ *gehören. Da* $11200 = 2^6 \cdot 5^2 \cdot 7$ *ist, kann für* e *nur eine Zahl gewählt werden, die nicht durch* 2, 5 *oder* 7 *geteilt werden kann, also z.B.* $e = 3533$*. Für die multiplikative Inverse* $e^{-1} = d$ *erhält man dann (und nur dann)* $d = 6597$*, weil* $3533 \cdot 6597 \equiv 1 \pmod{11200}$*.*

Bob hat also den öffentlichen Schlüssel $(3533, 11413)$ *und den privaten* $(6597, 11413)$*.*

Alice erhält den öffentlichen Schlüssel von Bob, entweder von ihm persönlich oder von einem authorisierten Schlüsselzentrum und verschlüsselt damit die Nachricht $m = 9726 < n = 11413$*. Sie berechnet also* $c = 9726^{3533} \bmod 11413 = 5761$ *und schickt* c *an Bob.*

Wenn Bob die Nachricht 5761 *erhält, so dechiffriert er sie mit seinem privaten Schlüssel und erhält wieder* $m = 5761^{6597} \bmod 11413 = 9726$.

Die Verschlüsselung und Entschlüsselung erscheint sehr rechenaufwendig. Es gibt aber effiziente Algorithmen, welche die Komplexität der Berechnung reduzieren können. Einige dieser Verfahren sind z.B. in [18] angeführt.

Die nachfolgende Abbildung 3.2 faßt den RSA-Algorithmus nochmals zusammen.

Erzeugung der Schlüssel

Erzeuge zwei große Primzahlen p, q

Berechne $n = pq$ und $\phi(n) = (p-1)(q-1)$

Wähle eine zufällige ganze Zahl e, $1 < e < \phi(n)$ mit $\gcd(e, \phi(n)) = 1$

Berechne d so, daß $ed \equiv 1 \pmod{\phi(n)}$

Öffentlicher Schlüssel: (e, n)

Privater Schlüssel: (d, n)

Chiffrierung

Nachricht: $m < n$

Chiffrierte Nachricht: $c = m^e \pmod{n}$

Dechiffrierung

Chiffrierte Nachricht: c

Nachricht: $m = c^d \pmod{n}$

Abbildung 3.2: Der RSA-Algorithmus

Literaturverzeichnis

[1] Berka, M., et al. *WWW – informační servery.* Unis Publishing s.r.o., Brno, ČR, 1997.

[2] Bogomolny, A. Interactive Mathematic Miscellany and Puzzles. http://www.cut-the-knot.com/algebra.shtml.

[3] Břehovský, P. *Praktický úvod TCP/IP.* Kopp, České Budějovice, ČR, 1994. ISBN 80-85828-18-9.

[4] Chapman, D. B., and Zwicky, E. D. *Building Internet Firewalls.* O'Reilly & Associates, Inc., Sebastopol, California, USA, 1995. ISBN 1-56592-124-0.

[5] Garfinkel, S., and Spafford, G. *Practical UNIX & Internet Security.* O'Reilly & Associates, Inc., Sebastopol, California, USA, 1996. ISBN 1-56592-148-8.

[6] Graham, R. L., Knuth, D. E., and Patashnik, O. *Concrete Mathematics: A Foundation for Computer Science.* Addison–Wesley Publishing Company, Reading, Massachusetts, USA, 1989. ISBN 0-201-55802-5.

[7] Hunt, C. *TCP/IP Network Administration.* O'Reilly & Associates, Inc., Sebastopol, California, USA, 1997. ISBN 1-56592-322-7.

[8] Laurie, B., and Laurie, P. *Apache: The Definitive Guide.* O'Reilly & Associates, Inc., Sebastopol, California, USA, 1999. ISBN 1-56592-528-9.

[9] Lewand, R. E. *Cryptological Mathematics.* The Mathematical Association of America, Washington, DC, USA, 2000. ISBN 0-88385-719-7.

[10] Lhotka, L. *Server v Internetu.* Kopp, České Budějovice, ČR, 1996. ISBN 80-85828-65-0.

[11] Mikle, P. *DHTML.* Unis Publishing s.r.o., Brno, ČR, 1997. ISBN 80-86097-09-9.

[12] Satrapa, P. *LINUX – Internet server.* Neokortex, s.r.o., Praha, ČR, 1996. ISBN 80-902230-0-1.

[13] Satrapa, P. *World-Wide Web pro čtenáře, autory a misionáře.* Neokortex, s.r.o., Praha, ČR, 1996.

[14] Satrapa, P. *WEB Design.* Neokortex, s.r.o., Praha, ČR, 1997. ISBN 80-902230-1-X.

[15] Šmrha, P. *Internetworking pomocí TCP/IP.* Kopp, České Budějovice, ČR, 1994. ISBN 80-85828-09-X.

[16] Stallings, W. *Network and Internetworking Security.* Prentice–Hall, Englewood Cliffs, New Jersey, USA, 1995. ISBN 0-7803-1107-8.

[17] Stallings, W. *Protect Your Privacy: The PGP User's Guide.* Prentice–Hall, Englewood Cliffs, New Jersey, USA, 1995. ISBN 0-13-185596-4.

[18] Stinson, D. R. *Cryptography: Theory and Practice.* CRC Press, Boca Raton, Florida, USA, 1995. ISBN 0-8493-8521-0.

[19] Štrupl, D. *JAVA pro každého.* PLUS Publishing, s.r.o., Praha, ČR, 1997. ISBN 80-85297-83-3.

[20] van der Lubbe, J. C. A. *Basic Methods of Cryptography.* Cambridge University Press, Cambridge, UK, 2000. ISBN 0-521-55559-0.

[21] Welsh, M., and Kaufman, L. *Running Linux.* O'Reilly & Associates, Inc., Sebastopol, California, USA, 1996. ISBN 1-56592-151-8.

[22] Zimmermann, P. R. *Official PGP User's Guide.* MIT Press, 1995. ISBN 0-262-74017-6.

Index