Linux
in Netzwerken

Burkhard Obergöker
Februar 2009

Inhalt

1 Warum Netzwerke

In der heutigen Zeit reicht es meistens nicht mehr aus, einen Computer als „Standalone“-Gerät zu benutzen. Selbst reine Home-PCs sind mittlerweile schon fast grundsätzlich ans Internet angeschlossen, was nichts anderes darstellt als die Integration eines Rechners in ein bestehendes Netzwerk. Unverzichtbar ist die Netzanbindung bei der Verwaltung von mehreren Computern, die z.B. in einem kleinen bis großen Unternehmen eingesetzt werden, allein schon um die Kommunikation zwischen den Mitarbeitern zu gewährleisten. Genauso kann auch der Freak zu Hause sein Netzwerk aufbauen, um z.B. Heretic mit mehreren Mitspielern über Netz zu spielen.

Nun reicht es aber nicht, einfach eine Netzwerkkarte einzubauen und die sie mit dem herumliegenden Kabel zu verbinden. Das Betriebssystem (egal, welches) muss erst noch konfiguriert werden, um Dienste in Anspruch zu nehmen, oder die eigenen anzubieten.

2 Netzwerk-Grundlagen

Als UNIX-Derivat ist Linux nicht nur grundsätzlich mit Netzwerkunterstützung ausgestattet, es ist auch auf ein vorhandenes Netzwerk angewiesen. Sehr viele Applikationen des Systems bestehen aus mehreren Modulen, die über Netzwerkfunktionen miteinander kommunizieren. Da aber nicht jeder Rechner mit einem Netzwerkanschluss ausgestattet ist, wird vom Betriebssystem ein Netzwerk „simuliert“, d.h. es wird vom Kernel ein (virtuelles) Gerät mit dem Namen ***lo0*** erzeugt, das wie eine Netzwerkkarte funktioniert, jedoch ausschließlich aus Software besteht. Auf diese Weise können z.B. die Netzwerkfunktionen überprüft werden, ohne dem Rechner einen echten Anschluss geben zu müssen.

Besitzt der Rechner tatsächlich einen Netzwerkzugang, so muss dieses ***Interface*** als Gerät im Verzeichnis „/dev“ bereitstehen und entsprechend konfiguriert werden. Diese Geräte werden nach einem bestimmten Schema benannt, das den Typ des Interfaces beinhaltet. So heißt die (erste) Ethernetkarte in einem PC „*/dev/eth0*“, eine Token Ring-Karte „*/dev/tr0*“ und ein Modem-Interface, das mit dem PPP arbeitet „*/dev/ppp0*“. Durch die Ziffern können mehrere Geräte gleichen Typs durch eine fortlaufende Nummerierung unterschieden werden, um sog. Multi Homed Gateways zu erstellen. Über diese Verfahrensweise können z.B. mehrere Netzwerke miteinander gekoppelt werden. In den jüngeren Linux-Derivaten werden diese Geräte nicht

Der Ursprung dieser Geräte liegt in den Kernel-Komponenten und müssen entweder fest in den Kernel gelinkt werden oder aber als Module zur Verfügung stehen. Erst wenn die Hardwareunterstützung korrekt läuft, kann die Konfiguration des Netzwerkes beginnen. Zunächst muss das Protokoll bekannt sein, das in dem Netzwerk benutzt werden soll. Bis auf wenige Ausnahmen ist Linux grundsätzlich auf das TCP/IP (Transmission Control Protocol/ Internet Protocol) angewiesen. Dieses Protokoll zeichnet sich dadurch aus, dass es auf fast jeder Art von Hardware läuft und eine große Menge unterstützender Tools anbietet.

2.1 TCP/IP

Zunächst ist ein Netzwerk nichts anderes als ein Zusammenschluss von mindestens 2 Computern, die über ein beliebiges Medium Daten austauschen können. Für diese Kommunikation muss eine Verständigungsform gefunden werden, nach der die betroffenen Rechner ihre Daten versenden und empfangen. Erst wenn beide dieselbe "Sprache" sprechen kann eine Kommunikation erfolgreich sein.

Eine solche Vereinbarung nennt sich "Protokoll" und ist als Standard festgelegt. Nun soll an dieser Stelle nicht der n-te Vergleich zwischen dem ISO-OSI-Referenzmodell und dem TCP/IP-

Schichtenmodell wiederholt werden, noch soll eine detaillierte technische Beschreibung des Paketaufbaus folgen, weil es über dieses Thema bereits gute und ausführliche Bücher gibt. Jedoch muss man sich die Struktur der einzelnen Protokollelemente vor Augen führen, um die Leistungsfähigkeit wie auch die Beschränkungen der hier beschriebenen Funktionen verständlich zu machen und deshalb soll dieses Kapitel einen pragmatischen Einblick in diese Technologie geben.

2.1.1 IP

In der untersten Schicht der für dieses Thema wichtigen Protokolle liegt das Internet Protocol, das IP:

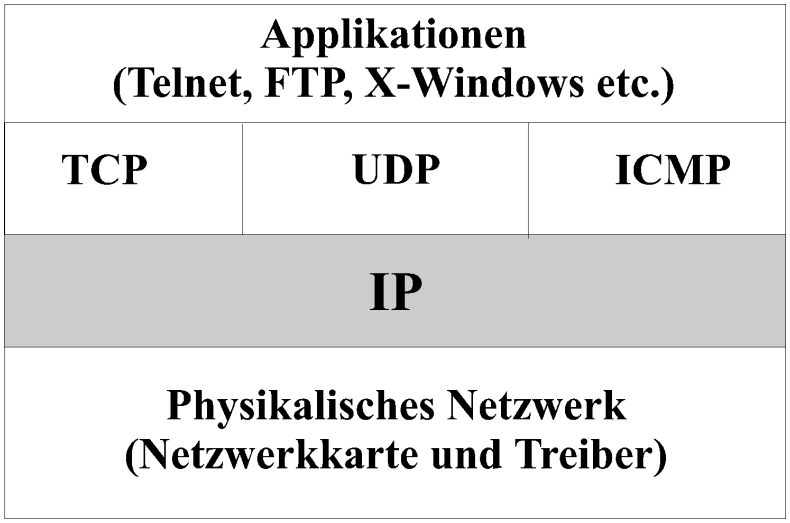


Abbildung 1: Position des IP

Wie in dem obigen Bild sichtbar, liegt es unmittelbar auf dem Netzkartentreiber und ist für die Vermittlung der Daten zuständig. Eine Funktion der Vermittlung ist die eindeutige Identifizierung der Kommunikationspartner, die in diesem Fall mit einer 4-Byte Nummer versehen werden. Allgemein werden diese IP-Adressen als Folge von 4 Zahlen dargestellt, die jeweils durch Punkte voneinander getrennt werden. Jede der 4 Zahlen kann einen Wert zwischen 0 und 255 annehmen, sodass - theoretisch - 232, also 4.294.967.296 Rechner adressiert werden können. Eine zulässige Adresse sähe z.B. so aus:

```
192.168.8.1
```

Zudem ist dieses System hierarchisch aufgebaut, d.h. die Adressen implizieren eine bestimmte Strukturierung der Knoten. Der vordere Teil einer IP-Adresse bezeichnet das Netzwerk, der hintere den Knoten in diesem Netz. Wo die Grenze zwischen Netz- und Knotenadresse liegt, ist über eine „**Netzmaske**“ festgelegt. wird die obige Adresse als Binärzahlen geschrieben, so sieht das so aus:

```
     192.     168.       8.       1       (Dezimal)
11000000.10101000.00001000.00000001       (Binär)
```

Die Netzmaske ist eine ähnlich aussehende Zahlenfolge, die überall dort, wo die Binärzahl in der Knotenadresse zum Netzwerk gehört, eine „1“ enthält. Eine gültige Maske zu der obigen Adresse wäre z.B.:

```
     255.     255.     255.       0       (Dezimal)
11111111.11111111.11111111.00000000       (Binär)
```

Nach dieser Maske würde also der Bereich „192.168.8“ zur Netzadresse und allein die letzte „1“ zum Knoten gehören. Alle Rechner in einem Netzwerk müssen dieselbe Netzadresse und

dieselbe Maske benutzen, um miteinander kommunizieren zu können. Dieser Verbund von Adresse und Maske wird meistens direkt hintereinandergeschrieben, do dass die obige Adresse mit „**192.168.2.1/255.255.255.0**“ oder in der Kurzschreibweise „**192.168.2.1/24**“.

Das folgende Bild zeigt eine gültige Kombination von Adressen in einem Netzwerk.

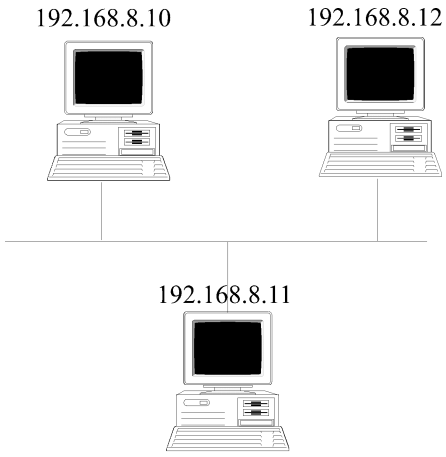


Abbildung 2: IP-Netzwerk

Ein solches Netz ist ein in sich abgeschlossenes System, das mit Knoten in anderen Netzen nur mit Hilfe von Vermittlern, den sog. „Routern“ oder „Gateways“ Kontakt aufnehmen kann.

In dieser Adressierung ist aber auch schon eine Zuordnung zu einem bestimmten Teilnetz vorhanden, da sie in 2 Teile gespalten wird: Der vordere Teil definiert das Netzwerk, der hintere den Rechner oder "Knoten" in diesem Netzwerk. Die Grenze zwischen beiden Teilen kann prinzipiell beliebig festgelegt werden, indem eine sog. "Maske" für diese Adresse festgelegt werden kann. Diese besitzt dieselbe Struktur wie die eigentliche IP-Adresse, jedoch ist bis zu einem bestimmten Grenze jedes Bit im vorderen Teil "1", wählend die hinteren den Wert "0" besitzen. Abwechselnde Folgen sind nicht zulässig. Wird eine Maske angegeben, so wird sie üblicherweise mit einem "/" von der Adresse getrennt, angehängt:

```
129.70.23.20/255.255.255.0
```

In dem obigen Fall würden die ersten drei Bytes das Netzwerk adressieren, während die hintere die jeweilige Knotennummer bestimmt. Bei allen Rechnern innerhalb eines Netzwerkes muss die Maske übereinstimmen, weil es sonst zu Missverständnissen kommt. Alternativ zu dieser Schreibweise kann die Maske auch als dezimale Zahl angehängt werden, die die Anzahl der auf "1" gesetzten Bits wiedergibt. Die obige Adresse könnte also auch so dargestellt werden:

```
129.70.23.20/24
```

Subnetze

Die Unterteilung in Netz- und Knotenadresse muss nicht unbedingt auf einer Byte-Grenze liegen, sofern der zu verwaltende Bereich es zulässt, darf dieser auch in Subnetze unterteilt werden, indem die beschriebene Grenze weiter nach rechts verlegt wird. In der obigen Form der Adressierung können max. 256 verschiedene Knotennummern angegeben werden (von denen 2 jeweils besondere Bedeutungen besitzen, s.u.). Soll das Netz z.B. in 2 Subnetze unterteilt werden, kann die Netzadresse im 1 Stelle nach rechts verschoben werden, sodass das obige Beispiel so aussehen würde:

```
129.70.23.20/255.255.255.128
```

bzw.

```
129.70.23.20/25
```

Zu beachten ist an dieser Stelle, dass die Knotenadressen nun nicht mehr von 1 bis 254, sondern in dem ersten Netz von 1 bis 126 und im zweiten von 129 bis 254 vergeben werden können. Da nun wiederum je 2 Adressen pro Subnetz reserviert werden, wird die Verlust an echten Knotenadressen umso größer, je mehr Subnetze gebildet werden.

Weiterhin gibt es in jedem Netz 2 Sonderfälle:

- Eine Knotenadresse mit dem Wert 0 bezeichnet keinen Rechner, sonder symbolisiert das Netzwerk als solches. Eine solche Adresse wird nicht wirklich versendet, sie ist jedoch für Konfigurationen besonders wichtig. Beispiel: 129.70.23.0/255.255.255.0
- Ist jedes Bit der Knotenadresse auf "1" gesetzt, stellt das einen "Broadcast", einen Rundruf an alle Rechner in dem aktuellen Teilnetz dar, der von allen Knoten gehört und ausgewertet wird. Beispiel: 129.70.23.255/255.255.255.0

Darüber hinaus gibt es eine Regelung, nach der bestimmten Adressenbereichen implizit Masken zugeordnet werden. Es handelt sich dabei um 4 Klassen von Netzadressen:

- **Klasse A:** 0.0.0.0 -126.255.255.255 mit Maske 255.0.0.0 bzw. /8
- **Klasse B:** 128.0.0.0 -191.255.255.255 mit Maske 255.255.0.0 bzw. /16
- **Klasse C:** 192.0.0.0 -223.255.255.255 mit Maske 255.255.255.0 bzw. /24
- **Klasse D:** 224.0.0.0 -255.255.255.255 mit Maske 255.255.255.255 bzw. /32 (reine HOST-Gruppe)

Achtung: Die Adressen 0.0.0.0, 127.0.0.0 und 255.255.255.255 können in der Praxis nicht verwendet werden, weil diese besondere Bedeutungen besitzen.

Bekommt die IP-Ebene von einer übergeordneten Schicht z.B. dem TCP einen Sendeauftrag, so nimmt es die empfangenen Daten als Block und "verpackt" sie in Protokollkopf, der unter anderem die Empfängeradresse, die Senderadresse und die Nummer des Transportprotokolls enthält, sodass die Gegenseite das Paket wieder "auspacken" und zuordnen kann. Durch die Unterteilung in Netz- und Knotenadresse kann das IP feststellen, ob der Zielpunkt noch im angeschlossenen oder in einem entfernten Netzwerk liegt. Im letzteren Fall muss dafür gesorgt werden, dass über bestimmte Mechanismen das Paket ggf. über mehrere Zwischennetze weitergeleitet wird. Daher besitzt das IP auch die Fähigkeit des Routings, was ein zwingende Voraussetzung für ein globales Protokoll ist.

Um besser mit den Adressen arbeiten zu können, wurden Synonyme für IP-Adressen eingeführt. Damit auch jeder Rechner weiß, welcher andere Rechner unter welchem Namen anzusprechen ist, gibt es eine Referenzdatei, die /etc/hosts, oder als Alternative den Domain Name Service (DNS), der in einem eignen Kapitel beschrieben wird.

Da das Internet unüberschaubar groß ist, muss von zentraler Stelle aus dafür gesorgt werden, dass jede dieser Adressen weltweit eindeutig ist. Es ist deshalb nicht zulässig, "irgendwelche" Adressen für ein eigenes Netzwerk zu verwenden, sofern es einen Zugang zum Internet besitzt. Entweder sind die eingesetzten über einen Provider eindeutig zugewiesen, oder es werden Adressen benutzt, die nach dem **RFC1918** nicht im Internet geroutet und damit offiziell nicht verwendet werden dürfen. Werden diese benutzt, ist sichergestellt, dass selbst bei einer bestehenden Verbindung zum Internet kein Schaden entstehen kann. Es handelt sich dabei um folgende Adress-Bereiche:

- Klasse A: 10.0.0.0 – 10.255.255.255

- Klasse B: 172.16.0.0 - 172.31.255.255
- Klasse C: 192.168.0.0 - 192.168.255.255

Diese Adressen können und dürfen dann auch für Testzwecke oder für "versteckte" Zwischennetze verwendet werden (s. Firewalling/Masquerading), ohne dass die ohnehin sehr knappen offiziellen IP-Adressen benutzt werden müssen.

2.1.2 TCP, UDP und ICMP

Die darüber liegende Ebene besteht aus unterschiedlichen Komponenten, die - gleichberechtigt und parallel - das IP als Kommunikationsmedium nutzen. Die für uns wichtigsten sind TCP (vom dem das TCP/IP seinen Namen bekommen hat) und das UDP, die beide der Transportschicht angehören.

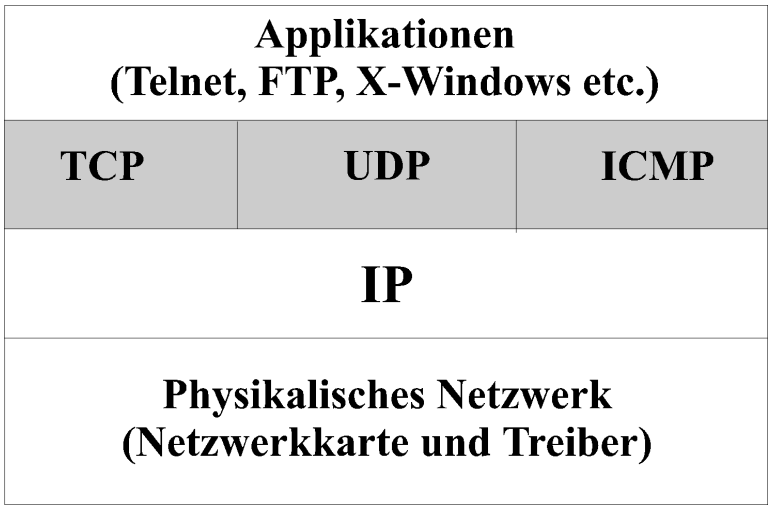


Abbildung 3: Position des TCP, UDP und ICMP

Auch diese beiden Protokolle bekommen nur eine Aufgabe: Die Daten mit den ihnen verfügbaren Mitteln an den Zielpunkt zu transportieren, wobei diesmal nicht der Zielknoten als Ganzes, sondern der Ort in diesem Rechner adressiert wird.

Das **Transmission Control Protocol (TCP)** stellt dabei mehr Funktionen zur Verfügung als das **User Datagram Protocol (UDP)**: Letzteres ist zwar schneller als das TCP, schickt aber die Daten ab, ohne sich darum zu kümmern, ob sie auch wirklich ankommen. Das UDP wird daher als "verbindungsloses" Protokoll bezeichnet. Das "verbindungsorientierte" TCP hingegen, schickt auf jede ankommende Meldung auch eine Bestätigung, ob und in welchem Zustand die Daten angekommen sind. Bleibt diese Bestätigung aus, so sendet der sendende Prozess seine Daten ein weiteres Mal, bis die Bestätigung eingeht, oder er davon ausgeht, dass die Verbindung unterbrochen ist.

Ähnlich wie die IP-Adressen werden auch in diesen Protokollen Zahlen für die Adressierung benutzt, die jedoch 16-Bit lang sind (also 0 bis 65535). Diese Adressen werden Ports genannt, also besteht die vollständige Adresse eines Prozesses, der einen Kommunikationskanal über TCP/IP benutzt, aus der IP-Adresse und der Portadresse, die in der üblichen Schreibweise durch einen Doppelpunkt getrennt hintereinander geschrieben werden (bekannt aus der URL-Schreibweise). Benutzt also ein Prozess auf der Maschine 129.70.32.11 den Port 4711, so lautet die korrekte Adresse 129.70.32.11:4711.

Allerdings ist diese Behauptung nicht ganz richtig, denn eine wichtige Adressierungs-Information fehlt hier noch, wird aber implizit mitgeliefert: Das Protokoll. Da das IP die Datenpaket je-

dem darüber liegenden Protokoll explizit zuordnen kann, ist der TCP-Port 4711 nicht derselbe wie der UDP-Port 4711. Auf diese Weise können sich die Transportprotokolle die Ports teilen, da sie selbst für deren Interpretation zuständig sind. Das ICMP hingegen benutzt keine Ports und ist daher immer eindeutig.

Da im Linux-System viele Prozesse (daemons) bereits gestartet sind, die darauf warten, von außen angesprochen zu werden, müssen dessen Portadressen auch bekannt sein. Zu diesem Zweck gelten die Portadressen von 1 bis 1024 als reserviert, und werden in einer zentralen Referenzdatei, der /etc/services, definiert. Die restlichen Portadressen von 1025 bis 65536 werden von Non-Standard-Daemons benutzt (z.B. Datenbankservern) oder dynamisch für die laufende Kommunikation vergeben.

Das **ICMP** ist ein Sonderfall unter den Transportprotokollen. Es wird nicht für den (Nutz-) Datentransport verwendet, sondern zur Fehleranalyse im IP-Netzwerk. Sie können von einer TCP- oder einer UDP-Nachricht ausgelöst werden und stellen das Äquivalent eines Return-Codes einer Funktion dar. Wann immer ein Problem bei der Vermittlung der Daten auftritt, sendet das IP eine ICMP-Rückmeldung. Auf diese Weise können Probleme entdeckt und gelöst aber auch einfache Laufzeitmessungen (ping) und Wegstreckenanalysen (traceroute) durchgeführt werden.

2.1.3 Routing

Ein einsames, isoliertes Netz ist solange ausreichend, wie nur eine recht kleine Menge von Computern miteinander kommunizieren müssen. Werden aber größere Einheiten, wohlmöglich noch geographisch voneinander getrennt, zur Kommunikation gezwungen, müssen Schnittstellen her, die die Daten auf unterschiedlichen Medien und über mehrere Knotenpunkte transportieren können. Zu diesem Zweck werden Router eingesetzt, die im allgemeinen Sprachgebrauch meistens als "Gateways" bezeichnet werden. Obwohl dieses nicht so ganz korrekt ist, kann aber davon ausgegangen werden, dass selbst in der korrekten Definition ein Gateway eine Obermenge der Funktionen eines Routers bietet, und somit die Bezeichnung akzeptabel ist.

Die Routingfunktionen sind fester Bestandteil des Linux-Kernels, sodass man prinzipiell jeden Linux-Rechner ohne sonderlichen Software-Einsatz als Router verwenden kann. Ein Router arbeitet ähnlich wie eine Bridge, lässt aber einen Protokollwechsel auf den unteren Ebenen zu. So kann auf dem Weg zum Ziel ein Datenpaket z.B. über ein Ethernetkabel, dann über eine Telefonleitung und schließlich über ein Glasfaserkabel gesendet werden, ohne das der Sender oder der Empfänger etwas davon bemerken. Dieser Vorteil bedeutet aber auch, dass mit diesem Mechanismus ausschließlich IP transportiert werden kann, andere Protokolle werden - wenn überhaupt - nur mit Zusätzen unterstützt.

Jeder Rechner, der sein Paket an einen Zielknoten senden möchte, das sich nicht in dem eigenen Netzwerk befindet, muss das Paket an den nächsten Router senden, weswegen alle verfügbaren Router im eigenen Netz bekannt sein müssen. Für diesen Zweck gibt es ausnahmsweise keine feste Konfigurationsdatei, weil die Routen durchaus während der Laufzeit geändert werden können. Referenz ist daher eine "Routing Table", die von jedem Netzknoten - unabhängig ob Router oder nicht - im Hauptspeicher gehalten wird.

Im TCP/IP-Bereich arbeitet ein Router grundsätzlich auf der Ebene des IP und ist daher in der Lage, die Netzadresse von der Knotenadresse zu unterscheiden, um daraus Schlussfolgerungen über den Weitertransport zu ziehen. Befindet sich der Zielrechner eines Paketes in einem angeschlossenen Netzwerk, so wird es direkt dorthin gesendet. Ist das nicht der Fall, wird in der Routing Table nachgesehen, welcher Router zum Weiterleiten in Frage kommt, und das Paket an ihn gesendet.

In kleineren Netzen ist das Routing trivial: Es existiert ein einziger Router, der alle Informationen in das Internet über den angeschlossenen Provider weitergibt. In diesem Fall kann mit statischen Routen gearbeitet werden, was bedeutet, dass als Voreinstellung der Router als "default Gateway" deklariert wird, sodass alles, das nicht in dem lokalen Netz bleiben soll, an diesen gesendet wird. Da sich diese Konfiguration nur sehr selten ändert, kann diese Route während des Boot-Vorganges gesetzt werden, ohne dass sie erneuert werden muss

Um größere Routen bestimmen zu können, werden Dienste verwendet, die die interne Routing-Table dynamisch verändern können. Zu diesen Diensten gehört z.B. der Daemon "routed", der seine Informationen über das RIP (routing information protocol) empfängt und weitergibt. Wird er gestartet, überprüft er zunächst alle angeschlossenen Netz-Interfaces, wobei er davon ausgeht, dass der Kernel für diese bereits das Forwarding übernimmt. Anschließend sendet er in jedes Netz ein Broadcast, der alle weiteren Router dazu veranlasst, ihm ihre eigenen Routing-Informationen zuzusenden.

Aus diesen Informationen wird dann die eigene Routing Table in einem festgelegten Intervall überprüft und aktualisiert. Durch den Automatismus dieses Dienstes ist es in den meisten Fällen nicht notwendig, bestimmte Routen manuell in eine Konfigurationsdatei einzutragen, trotzdem bleibt die Option der manuellen Konfiguration, einfach aus dem Umstand heraus, dass Störungen auftreten können oder sonstige Zugriffe nötig werden.

3 Konfiguration von Netz-Interfaces

Je nach Distribution stehen verschiedene Mittel zur Verfügung, um die Netzanschlüsse zu konfigurieren. Bei der Distribution von S.u.S.E. geschieht dieses durch das zentrale Konfigurationstool „YaST“, das mit dem nachfolgend abgebildeten Dialog eine interaktive und übersichtliche Konfiguration zulässt.

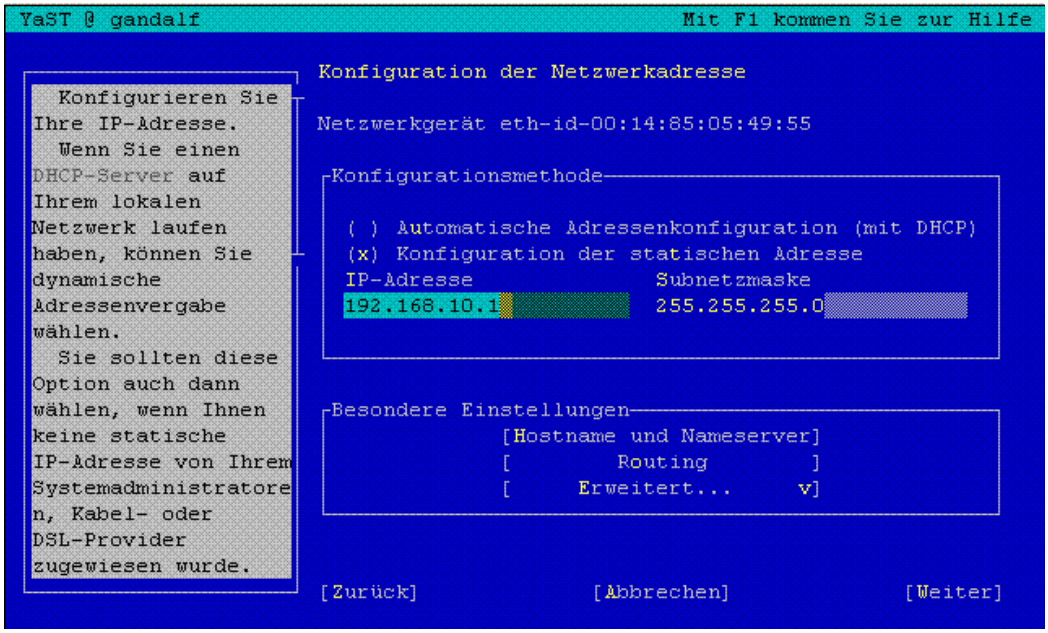


Abbildung 4: Netzwerk-Adresse im SuSE-YAST

Um unabhängig von den Distributionen werden zu können, ist es wichtig, sich über die Hintergründe klar zu werden. Der Kommandozeilenbefehl lautet zu diesem Zweck „**ifconfig**“, mit dem zur Laufzeit ein Interface beliebig oft (ohne reboot!!) verändert werden kann. Der allgemeine Aufruf erfolgt grob nach diesem Schema (s. „man ifconfig“):

```
ifconfig interface [aftype] options | address ...
```

Wenn also nach dem obigen Beispiel ein Netzanschluss über die Netzwerkkarte „eth0“ mit de Adresse 192.168.8.10 hergestellt werden, dann lautet der richtige Befehl

```
ifconfig eth0 192.168.8.10 netmask 255.255.255.0
```

mit diesem Befehl wird das Device auch gleichzeitig aktiviert, was sonst mit dem Befehl

```
ifconfig eth0 up
```

geschehen müsste. Das Gegenstück zu dem Schalter „up“ ist konsequenterweise „down“, um ein Interface während der Laufzeit logisch vom Netzwerk zu trennen.

Sofern das Netz kein in sich abgeschlossenes System ist, muss die Schnittstelle zur „Außenwelt“ definiert werden, was im einfachsten Fall ein einziger Rechner ist, ein „Router“. Ein solcher Router muss nicht nur eine Netzwerkadresse im lokalen Netz besitzen, sondern zusätzlich eine im weiterführenden Netzwerk (üblicherweise das des IN-Providers). Da dieser alle Daten erhalten und weiterreichen soll, die ihren Zielpunkt nicht im lokalen Netz besitzen, wird er als „default route“ eingetragen, wobei die Adresse im lokalen Netz ausschlaggebend ist.

Sei beispielsweise der Router in unserem Netz 192.168.8.0 der Rechner mit der Nummer 192.168.8.1, so wird diese Route mit dem folgenden Befehl im System eingetragen:

```
route add default gw 192.168.8.1
```

Aufwendiger wird diese ganze Konstruktion, wenn mehrere Netzwerke durch mehrere Router verbunden werden. In diesem Fall muss einem Knoten genau mitgeteilt werden, welches Netz über welchen Router erreicht werden kann. Auch hier wird der „route“-Befehl benutzt, wobei das Schlüsselwort „default“ mit der Adresse des zu erreichenden Netzes ersetzt wird.

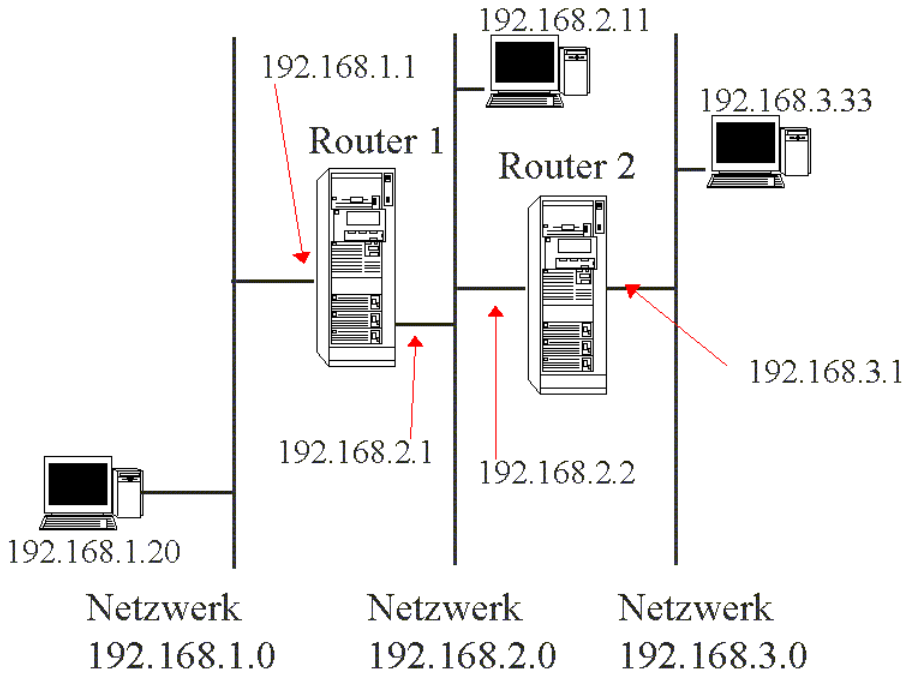


Abbildung 5: Routing

Die Struktur im obigen Bild stellt 3 Netzwerke dar, die durch 2 Router gekoppelt werden. Diese Router besitzen zwei Netzwerkkarten und daher auch 2 IP-Adressen. Solche Rechner werden auch als „**Multihomed Hosts**“ bezeichnet. Der Rechner mit der Adresse 192.168.2.11 muss also Kenntnis von beiden Routern haben, um beide Netze erreichen zu können. Die benötigten Befehle lauten deshalb:

```
route add -net 192.168.1.0 gw 192.168.2.1
route add -net 192.168.3.0 gw 192.168.2.2
```

Die anderen beiden abgebildeten Rechner „sehen“ jeweils nur einen Router, über den sie Zugang zu allen weiten Netzwerken bekommen können, weshalb bei ihnen ein „default“-Eintrag genügt. Um eine vollständige Kommunikation herzustellen, müssen alle beteiligten Knoten von allen **direkt** erreichbaren Routern Kenntnis besitzen. Router in nur mittelbar erreichbaren Netzen dürfen in einer Routingtabelle nicht auftauchen. Es wäre also unzulässig, dem Rechner mit der Nummer 192.168.1.20 den Router 192.168.2.1 einzustellen.

Es reicht auch nicht, wenn nur ein Kommunikationspartner die Route zum Ziel kennt, der Andere muss auf dieselbe Art antworten und benötigt daher auch die vollständige Wegstrecke. wenn also eine Kommunikation zwischen zwei Rechnern nicht stattfindet, sind immer die Routingeinträge **beider** Partner zu untersuchen. Das beste Werkzeug für einen Streckentest ist das Werkzeug „ping“. Dieses Programm sendet ein IP-Paket zu dem angegebenen Rechner mit der Aufforderung dasselbe zu beantworten. Die Fehlermeldungen sind dabei sehr detailliert, so dass es wesentlich mehr Aufschluss über die Fehlerquelle gibt als der Start des eigentlich benötigten Programms. Erzeugt der Aufruf des auf dem PC 192.168.1.10 ausgeführten Befehls

```
ping 192.168.3.33
```

die Fehlermeldung „*network unreachable*“, so kann davon ausgegangen werden, dass ein Routingeintrag auf dieser Strecke irgendwo falsch gesetzt wurde.

Ein Linux-PC, der lediglich per DFÜ zu einem Server eines ISPs an das Internet angeschlossen ist, reicht eine default-Route aus, schließlich ist der einzige sichtbare Router der angewählte Server.

Die Router selbst – die schließlich auch auf dieselbe Art der Kommunikation angewiesen sind – benötigen meistens eine detailliertere Beschreibung der erreichbaren Netzwerke. Diese Tabellen bei jeder Veränderung immer manuell zu korrigieren ist aber eine praktisch nicht zu leistende Aufgebe. Daher gibt es einen Dienst, der in der Lage ist, durch Abfrage der umliegenden Netzknoten eine gültige und vollständige Routingtabelle zusammenzustellen. Dieser „**routed**“ benutzt ein separates Protokoll, das RIP (routing information protocol), um seine Informationen zusammenzustellen. Ein „frisch“ hochgefahrener Router benötigt daher eine gewisse Zeit, um den angeschlossenen Clients alle Wege ins Internet wieder bereitstellen zu können. Trotzdem ist dieses Verfahren wesentlich stabiler und effektiver als eine manuell geführte Liste.

All diese Befehle behalten aber ihre Gültigkeit ausschließlich nur solange, wie das System läuft. Bei einem Neustart müssen alle Konfigurationsbefehle wiederholt werden, um die Netzanbindung herzustellen. Daher werden diese in einem Startup-Skript der Reihe nach ausgeführt.

4 Internetanbindung über DFÜ

Anders als die beschriebene Art der Netzwerkkopplung erfolgt die Konfiguration einer DFÜ-Anbindung. Für eine Kopplung über ein Modem steht der Dienst „pppd“ zu Verfügung. Dieser Dämon erzeugt eine Verbindung über ein Modem und erstellt automatisch die benötigten Interfaces, Adressen und Routen im System. Für die Erstellung einer ISDN-Verbindung existiert ein anderes Paket das sich „i4l“ (ISDN for Linux) nennt. Beide sind abhängig von er vorhandenen Hardware und dem unterstützten Providerprotokoll einzurichten, so dass an dieser Stelle tiefergreifende Literatur zu Rate gezogen werden sollte. Die meisten Distributionen bieten aber auch schon interaktive Konfigurationswerkzeuge an, mit denen auf einfache Weise eine Internetanbindung ermöglicht wird. Soweit die grafische Oberfläche „X-Windows“ und KDE eingerichtet sind, kann „kisdn“ bzw. „kppp“ benutzt werden, die üblicherweise beide in der Softwaresammlung zu dem KDE enthalten sind.

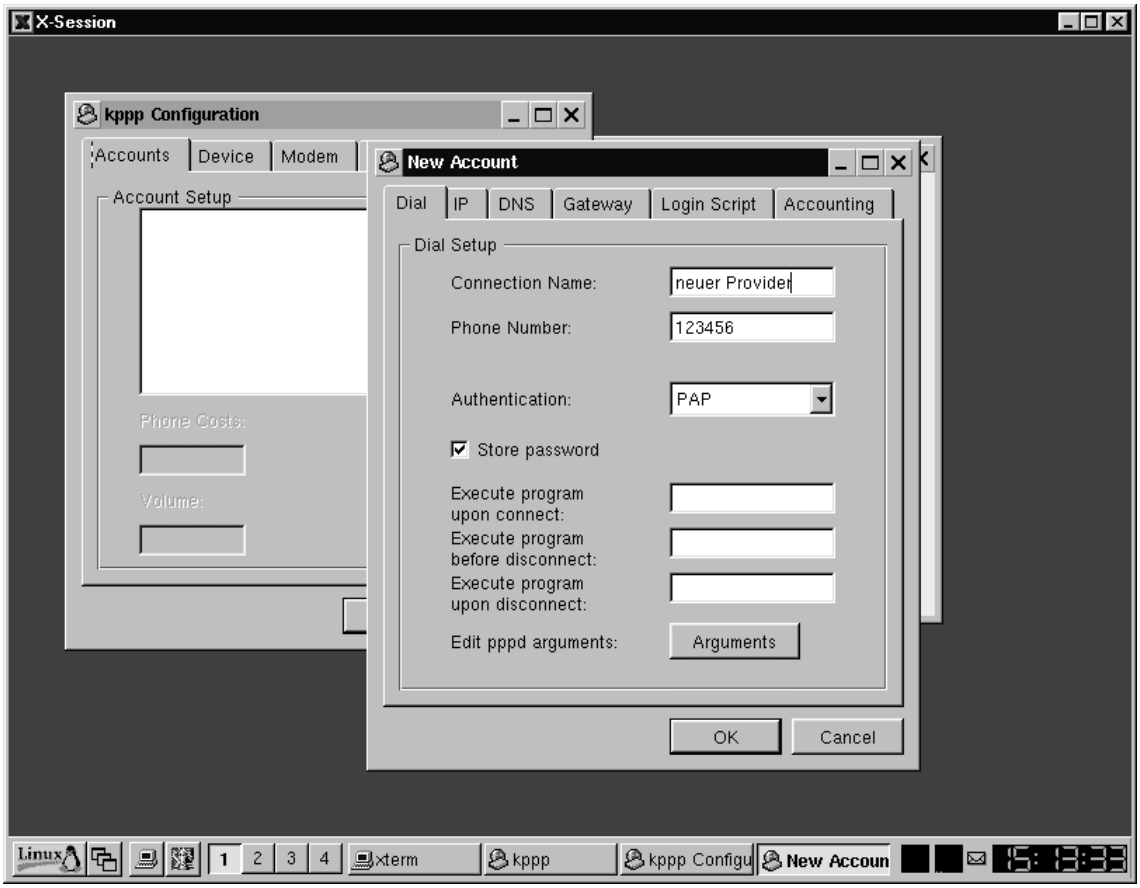


Abbildung 6: Konfiguration im Kppp

Es gibt also keinen zentralen Menüpunkt im Linux-System, in dem die DFÜ-Anschlüsse konfiguriert werden. Da diese Methoden nicht nur undurchsichtig, sondern auch extrem abhängig von der Distribution und der installierten optionalen Software sind, soll an dieser Stelle genauer auf die jeweilige Konfiguration eingegangen werden.

4.1 PPP

Das PPP (Point to Point Protocol) basiert auch auf der Verbindungserstellung zweier Knoten, mit dem Vorteil, dass es mehrere Protokolle "durchschleusen" kann. So kann nicht nur eine TCP/IP-Verbindung, sondern auch SPX/IPX (Novell) übertragen werden. Der von den meisten Linux-Distributionen mitgelieferte ppp-Daemon ist der "pppd", wie er von Al Longyear (longyear@pobox.com) für Linux portiert wurde und unter dieser Adresse zu beziehen ist:

ftp://cs.anu.edu.au/pub/software/ppp/.

Der pppd benutzt die im Kernel eingebauten PPP-Fähigkeiten und ist daher ebenfalls sehr stabil und schnell. Normalerweise können die ppp-Dienste gleich mit der Linux-Installation automatisch installiert werden. Sollte es aber nicht so sein, muss die Installation "von Hand" durchgeführt werden.

4.1.1 Kompilierung

Zunächst muss dafür gesorgt werden, dass im Kernel das ppp als Modul (!!) eingebaut ist. Zu diesem Zweck müssen diese Parameter im "Network device support" der Kernel-Konfiguration gesetzt worden sein:

```
<M> PPP (point-to-point) support
```

```
<M> SLIP (serial line) support
[*]  CSLIP compressed headers
```

Ist das tgz-Paket des pppd entpackt, muss in dem neu entstandenen Verzeichnis zunächst

```
./config
```

und danach

```
make kernel
```

gestartet werden. Durch den letzten Befehl wird das vorliegende Modul "pppd.c" in den Kernel eingepflegt. Das bedeutet natürlich auch, dass an dieser Stelle zunächst der Kernel neu kompiliert und anschließend das System mit diesem neuen Kern gestartet werden muss. Dann erst kann im ppp-Verzeichnis mittels der Befehle

```
make
```

und

```
make install
```

(in dieser Reihenfolge) das neue Modul in dem entsprechenden Verzeichnis "eingebaut" werden.

4.1.2 Statische Verbindung

Vor dem Erstellen der entsprechenden Skripten für den Verbindungsaufbau werden folgende Informationen benötigt:

- Die Telefonnummer des anzurufenden point-to-point-Partners,
- das eigene Login,
- das zugehörige Passwort

Und zusätzlich noch:

- DNS des Providers

Als DNS muss nicht zwingenderweise der des Providers benutzt werden, es kann auch ein beliebiger DNS sein, solange er korrekte Informationen über das gesamte Internet liefert. Sollte der Provider allerdings seine eigene Netzwerkstruktur mittels Firewall schützen, bleibt diese Wahl nicht.

Zunächst wird das Startskript für den pppd erstellt, in der die Parameter für den Verbindungsaufbau festgelegt werden. Eine solche Datei wird im Paket pppd mitgeliefert und trägt den Namen "ppp-up". Üblicherweise wird sie durch den install-Befehl automatisch in das Verzeichnis "/etc/ppp" kopiert:

```
#!/bin/sh
#
# /etc/ppp/ppp-up
#
# Aufbau einer PPP Verbindung
#
localip=0.0.0.0
remoteip=
device=/dev/modem
pppflags="38400 modem defaultroute"
/usr/sbin/pppd lock connect \
     '/usr/sbin/chat -f /etc/ppp/ppp.chat' \
     $device $pppflags $localip:$remoteip
```

Da beim Einwählen beim Provider normalerweise dynamische Adressen vergeben werden und daher weder die eigene noch die entfernte IP-Adresse vorhersagbar ist, wird für die lokale Adresse "0.0.0.0" eingetragen, was implizit bedeutet, dass die reale Adresse erst während des Verbindungsaufbaus ermittelt wird. Die Adresse des PPP-Partners bleibt dann völlig leer.

Das Schlüsselwort "defaultroute" veranlasst den pppd nach der Ermittlung der entfernten Adresse diese als "default route" einzutragen, damit auch wirklich alle Datenpakete an diese Adresse weitergegeben werden.

Zum Aufbau einer PPP-Verbindung wird eine bereits funktionierende physikalische Verbindung vorausgesetzt. Das bedeutet, dass ppp nicht selbst das Wählen eines Modems steuern kann. Für diesen Zweck wird "chat" eingesetzt, das in der Lage ist, mit dem Kommandomodus des Modems umzugehen.

In dem vorigen Skript verdient der Parameter "connect" eine besondere Beachtung. Er verlangt als Wert ein Shell-Kommando, das dafür sorgt, dass der pppd eine physikalisch vollständige Verbindung vorfindet. Er ist natürlich obsolet, wenn eine Standleitung existiert, in der ohnehin schon ein anderer pppd auf ein Gegenüber wartet. Das hier verwendete Programm "chat" steuert die Kommunikation mit dem Modem, wobei jeweils ein zu erwartender String vorgegeben wird und anschließend der zu übermittelnde String in derselben Zeile folgt. In der nächsten Zeile kann wiederum ein zu erwartender String folgen usw. Weiterhin können "KO-Kriterien mitgegeben werden. Es handelt sich dabei wiederum um Zeichenketten, bei deren Auftreten chat sich mit einer Fehlermeldung beendet. Diese "KO-Kriterien" werden durch das vorangestellte Schlüsselwort "ABORT" gekennzeichnet.

Ein Beispielskript, das für den "chat" benötigt wird, liegt im ppp-Paket unter dem Namen "ppp.chat" vor, und wird ebenfalls im Verzeichnis "/etc/ppp" abgelegt. Vorsicht: Da diese Datei das eigene Passwort im Klartext enthält, muss sie dringendst vor den Lesezugriffen Anderer geschützt werden. (Befehl: chmod 600 ppp.chat).

```
TIMEOUT 90
ABORT "NO CARRIER"
ABORT BUSY
ABORT "NO DIALTONE"
ABORT ERROR
"" ATZ
OK ATDT<Provider-Nummer>
CONNECT ""
ogin: <Login-Kennung>
word: <Passwort>
```

Die Nummer des Providers muss zwingenderweise so eingegeben werden, wie es das Modem verlangt. Ist also beispielsweise die Telefonnummer "12345" und das Modem ist direkt an der öffentlichen Telefonleitung angeschlossen, so lautet die Zeile

```
OK ATDT12345
```

Ist das Modem an einer Nebenstelle angeschlossen, an der mit "0" eine Amtsleitung geöffnet wird, so sollte die Nummer so eingetragen werden:

```
OK ATX0DT0w12345
```

Für weitere Einstellungen sollte jedoch das Modem-Handbuch konsultiert werden, da hier nicht alle möglichen Einstellungen berücksichtigt werden können Login und Passwort werden ohne Anführungszeichen, aber durch ein Leerzeichen von dem vorangegangenen String getrennt angefügt. In dem Skript dürfen weder Kommentare noch Leerzeilen eingefügt werden.

Falls die Einwahl nicht funktioniert, sollte mit einem Terminal-Programm (z.B.) "minicom" manuell überprüft werden, was denn nun über die Leitung gesendet und erwartet wird. Wenn

nach der Eingabe des Login/Passwortes undefinierbare Zeichen erscheinen, ist die Gegenstelle schon dabei, eine Verbindung auszuhandeln, worauf der pppd dann aufbauen kann.

Die beschriebene Art des Verbindungsaufbaus kann allerdings nur glücken, wenn die Gegenseite ein UNIX-like Login verlangt. Wenn es sich aber z.B. um ein Microsoft-System handelt, muss ein anderer Weg der Identifizierung gewählt werden. Zu diesem Zweck ist der pppd in der Lage, ein solches System zu erkennen und entsprechend darauf zu reagieren.

Eine mögliche Erweiterung der Authentifizierung ist das **PAP (User/Password Authentication Protocol)**, das ähnlich wie das oben beschriebene Login-Verfahren funktioniert. Auch hier wird vom Server Login und Passwort abgefragt, jedoch erwartet anschließend die Client-Seite ebenfalls eine Authentifizierung des Servers. Um vorzugeben, was erwartet und übermittelt wird, dient die Datei /etc/ppp/pap-secrets, die fast identisch mit der folgend beschriebenen Datei ist, die für den CHAP-Aufbau zuständig ist.

Windows NT erwartet eine Identifikation auf verschlüsselter Art. In diesem Fall wird nicht wie bei einem Login der String "Login" im Klartext versendet und vor allem nicht das Passwort. Dieses Verfahren nennt sich **Challenge/HandshakeAuthentication Protocol (CHAP)**. Zu diesem Zweck wird im Verzeichnis /etc/ppp eine Datei namens "chap-secrets" benötigt, deren Inhalt die zulässigen user/password-Informationen enthalten sind. Da auch diese im Klartext abgelegt sind, gilt auch hier, die Datei entsprechend zu sichern.

Achtung: Durch die Verschlüsselung der Passwörter wird eine Sicherheit suggeriert, die de facto nicht existiert. Tatsächlich kann ein Angreifer das verschlüsselte Passwort zusammen mit dem Login so wie es ist als Zugangsberechtigung verwenden!

Für die Datei chap-secrets müssen 3 Dinge bekannt sein: Der Name des Dialin-Servers, der eigene Rechnername und das Passwort, das vom dem Provider mitgeteilt wurde. Die Namen der Rechner sollten üblicherweise deren IP-Namen entsprechen. Da diese häufig nicht einmal bekannt sind, können als Alternative dem pppd die Parameter "name" und "remotename" mitgegeben werden. In letzterem Fall müssen allerdings die Namen der chap-secrets mit den vergebenen übereinstimmen und außerdem der Wert des Parameters "name" dem vom Provider mitgeteilten Login-Namen. Anstatt den pppd mit zusätzlichen Parametern zu starten, kann auch in die **/etc/ppp/options** folgende Zeilen eingefügt werden:

```
remotename <RemoteHost>
name <UserName>
```

Nun müssen 2 Zeilen in die chap-secrets eingefügt werden:

```
<RemoteHost> <UserName> "<Password>"
<UserName> <RemoteHost> "<Password>"
```

Zu beachten ist, dass in der 2. Zeile derselbe Inhalt nur in unterschiedlicher Reihenfolge abgelegt wird, da prinzipiell der angewählte Rechner sich mit einem anderen Passwort identifizieren kann, als der wählende Rechner (entspricht aber nicht gängiger Praxis). Auch muss das Passwort in doppelte Anführungszeichen eingetragen werden, da sonst die Übermittlung des Passwortes nicht korrekt durchgeführt werden kann.

Sollte die Verbindung aufgrund eines falschen Servernamens nicht klappen, so lohnt ein Blick in die Datei "/var/log/messages", in der eine Zeile wie die folgende auf den richtigen Namen schließen lässt:

```
Jul 22 20:37:30 gandalf pppd[993]: rcvd [CHAP Challenge id=0xc1 <84c-
c70f79d79bd7d59ae7f5c0f74c2a5>, name = "HANDI4"]
Jul 22 20:37:30 gandalf pppd[993]: No CHAP secret found for authenti-
cating us to HANDI4
```

Der gesuchte Name ist eindeutig "HANDI4"

Der Ablauf der Identifikation ist dann bei PAP und CHAP ähnlich: Zuerst wird über CHAT die Verbindung mit dem Dialin-Rechner aufgenommen, dann wird in der secrets-Datei der Eintrag gesucht, in dem der wählende Rechner sein eigenen Namen findet (IP-Name oder "name"-Wert aus den Parametern). Das in dieser Zeile befindliche Passwort wird dem Server dann vermittelt. Dieser schickt dann seinerseits ein Passwort, das dann mit dem verglichen wird, das sich in der korrespondierenden Zeile befindet. PAP und CHAP sind also keine Alternativen für den Verbindungsaufbau des CHAT-Programmes, weil dieses immer noch mindestens für die Anwahl benötigt wird. Normalerweise fällt jedoch das Login/Passwort Fragespielchen weg, sodass man diese Zeilen aus dem CHAT-Skript entfernen muss, wonach es beispielsweise so aussehen kann:

```
TIMEOUT 90
ABORT "NO CARRIER"
ABORT BUSY
ABORT "NO DIALTONE"
ABORT ERROR
"" ATZ
OK ATDT<Provider-Nummer>
CONNECT ""
```

Zum Verbindungsabbau kann natürlich mittels "ps ax" die PID des pppd herausgesucht werden und dieser dann mit "kill" beendet werden, doch ist es einfacher, dazu ein Skript zu benutzen, das mit dem pppd-Paket mitgeliefert wird und "ppp-down" heißt:

```
ppp-down
#!/bin/sh

DEVICE=ppp0

#
# If the ppp0 pid file is present then the program is running.
Stop it.
if [ -r /var/run/$DEVICE.pid ]; then
     kill -INT `cat /var/run/$DEVICE.pid`
#
# If unsuccessful, ensure that the pid file is removed.
#
     if [ ! "$?" = "0" ]; then
           echo "removing stale $DEVICE pid file."
           rm -f /var/run/$DEVICE.pid
           rm -f /usr/spool/uucp/LCK..cua1
           exit 1
     fi
#
# Success. Terminate with proper status.
#
     echo "$DEVICE link terminated"
     exit 0
fi
#
# The link is not active
#
echo "$DEVICE link is not active"
exit 1
```

In diesem Skript wird mit Hilfe der PID-Datei der pppd wieder gestoppt. Während des Starts schreibt der pppd seine eigene Prozessnummer in das Verzeichnis "/var/log" Dieses wird verwendet, wenn das Kill-Signal gesendet wird.

4.1.3 Dial-on-Demand

Der Nachteil der oben beschriebenen Verbindung ist die fehlende Flexibilität. Die Verbindung muss durch den root initiiert werden und bleibt solange aktiv, bis sie durch denselben wieder abgebaut wird. Bei einer Telefonverbindung würde das unnötige Kosten nach sich ziehen.

Eleganter ist es, dafür zu sorgen, dass die logische Verbindung zwar sofort erstellt wird, die physikalische aber erst dann, wen tatsächlich ein Datenpaket an den anderen Knoten gesendet werden soll. Das pppd ist seit der Version 2.3.3 dazu in der Lage, eine solche "dial on demand" Verbindung herzustellen. Wird in den Parametern des pppd das Schlüsselwort "demand" eingefügt, wird so verfahren, wie oben beschrieben. Der Parameter „idle“ gibt in Minuten an, nach welcher Zeitspanne „Stille“ auf der Leitung der pppd die Verbindung unterbricht. Sobald wieder Daten über die Leitung gesendet werden sollen, stellt der pppd die Verbindung wieder her.

```
#!/bin/sh
#
# /etc/ppp/ppp-up
#
# Aufbau einer PPP Verbindung
#
localip=195.180.193.233
remoteip=195.180.193.6
device=/dev/modem
pppflags="38400 modem defaultroute demand idle 120 \
ipcp-accept-local ipcp-accept-remote"
/usr/sbin/pppd lock connect \
     '/usr/sbin/chat -f /etc/ppp/ppp.chat' \
     $device $pppflags $localip:$remoteip
```

Es gibt an dieser Stelle nur ein Problem zu überwinden: Da der pppd nach dem Start zwar keine Verbindung, jedoch ein Netz-Interface und ggf. eine Route einrichtet, muss er die eigene IP-Adresse und die des angerufenen Rechners kennen. Eine feste Adressenzuordnung ist aber sehr teuer und wird bei den meisten Providern für Privatanschlüsse nicht eingerichtet.

Man kann den Provider aber in gewisser Weise überlisten, indem man einfach die vorhandenen und vergebenen Adressen ermittelt. Zu diesem Zweck wird eine PPP-Verbindung nach obigem Beispiel gestartet. in der Datei "/var/log/messages" erscheinen dann im Klartext die zugewiesenen Adressen:

```
Jul  3 21:35:55 gandalf pppd[682]: local  IP address 195.180.193.233
Jul  3 21:35:55 gandalf pppd[682]: remote IP address 195.180.193.6
```

Der Eintrag "remote IP address" signalisiert die Adresse des Dial-in Servers. Sofern der Provider nicht mehrere Rechner für die Einwahl bereitgestellt hat und zusätzlich seine Telefonanlage die Verbindungen dynamisch auf beide verteilt, kann angenommen werden, dass diese bei jedem Verbindungsaufbau identisch ist.

Der "local IP address"-Eintrag stellt die eigene, vom Server zugewiesene Adresse dar, die bei jeder Verbindung anders aussehen kann. Daher ist es relativ egal, welche Adresse hierfür gewählt wird, sofern sie eine von den verfügbaren Adressen ist. Die beiden Einträge "ipcp-accept-local ipcp-accept-remote" im ppp-Skript sorgen dafür, dass der einwählende Rechner nicht auf den vorgegeben Adressen besteht, sondern die Vorgaben des Servers übernimmt. Doch grundsätzlich versucht er erst einmal, die eigenen mit dem Server abzugleichen. In vielen Fällen (wenn die lokale Adresse nicht vergeben ist , und die ppp-Software des Servers dies unterstützt), werden die Anfragen des Clients übernommen und so eine quasi-stabile IP-Adresse vergeben.

Wenn aber nun trotz allem die IP-Adresse des eigenen Rechners verändert wird, geht das erste IP-Paket allerdings üblicherweise verloren, weil der Absender darin vermerkt wird. Es landet

dann irgendwo im Internet-Nirwana, mut der Folge, dass die entsprechende Applikation (z.B. der Netscape) einmal unterbrochen und die Verbindung erneut aufgebaut werden muss. Weiterhin kann der Aufbau der Verbindung nur vom internen Netz aus initiiert werden und nicht von außen.

Bei kurzzeitigen Verbindungen ist diese Art des Verbindungsaufbaus unkritisch. Wird aber z.B. eine ftp-Verbindung auf diese Weise unterbrochen und mit einer anderen Adresse fortgesetzt, bedeutet das das Ende der Session. Für solche Fälle muss also eine statische Verbindung aufgebaut werden, oder (noch einfacher) ein "Ping" auf die Zielstation solange aktiv gehalten werden, bis der komplette Transport abgeschlossen ist. Ping erzeugt so gut wie keine Netzlast, sorgt aber dafür, dass die ppp-Verbindung geöffnet bleibt.

Dem Problem des verlorenen ersten Paketes kann allerdings entgegengewirkt werden, indem ein relativ neuer **Dynamic-IP-Patch** aktiviert wird, der im **Kernel 2.2.x** enthalten ist. Dieser Patch kann zur Laufzeit eingeschaltet werden und versucht - sofern möglich - den Absender eines IP-Headers bei dem Wechsel der Interface-Adresse zu überschreiben. Offensichtlich funktioniert der Patch lt. Autor Juan Jose Ciarlante noch nicht zufrieden stellend, doch erweist er sich in der Praxis als erstaunlich stabil. Die Aktivierung des Patches übernimmt der Befehl:

```
echo "7" > /proc/sys/net/ipv4/ip_dynaddr
```

Ein Tipp noch am Rande: Wenn in der /etc/resolv.conf als DNS eine externe Adresse angegeben ist (z.B. der DNS des Providers), so wird bei jeder möglichen und unmöglichen Gelegenheit dieser DNS befragt. D.h. es wird jedes Mal, wenn ein Rechnername in eine IP-Adresse umgewandelt werden soll, eine ppp-Verbindung aufgebaut, was natürlich nicht notwendig ist, wenn ein lokales Netz angeschlossen ist und die Kommunikation nur in diesem stattfinden soll. Für diese Situation lohnt es sich, einen eigenen DNS aufzusetzen, der nur die Knoten des lokalen Netzes und einen "Forwarder"-Eintrag auf den entfernten DNS enthält. Auf diese Weise wird nur dann eine Telefonverbindung geöffnet, wenn ein Rechner außerhalb des eigenen Netzes angesprochen werden soll.

Da die gerade beschriebene Art des Verbindungsaufbaus nur dann initiiert wird, wenn auch wirklich Daten gesendet werden, lohnt es sich, diese beim Bootvorgang automatisch starten zu lassen. Zu diesem Zweck muss ein Start/Stop-Skript in den Boot-Vorgang eingebaut werden, für das in SYSV-konformen Distributionen ein separates Startup-Verzeichnis existiert (z.B. "/etc/rc.d" bei Red Hat und "/sbin/init.d" bei S.u.S.E.). Um es während des Start und Stopp-Vorganges zu aktivieren, wird zusätzlich je ein Symbolischer Link in dem jeweiligen Level-Verzeichnis benötigt, das die jeweilige Distribution vorgibt. Die Position des Startskriptes sollte unmittelbar nach dem Aufbau des Netzwerkes, aber vor den Netzdiensten wie Firewall, Masquerade etc. erfolgen, weil Teile dieser Dienste auf die Existenz der (logischen) Netz-Interfaces angewiesen sind. Vorausgesetzt, dass das Startskript /etc/ppp/ppp-up und das Stop-Skript /etc/ppp/ppp-down heißt, könnte die rc-Datei so aussehen:

```
case "$1" in
    start)
        echo -n "Starting pppd: "
        /etc/ppp/ppp-up
    ;;
    stop)
        echo -n "Shutting down pppd: "
        /etc/ppp/ppp-off
     ;;
```

4.2 ISDN

ISDN ist mittlerweile ein recht diffuses Kapitel für Linux, weil viele Treiber, noch mehr Patches aber recht wenig vollständige Dokumentation zu diesen Modulen existieren. Die hier angegebenen Beschreibungen beziehen sich auf die Kernelversion 2.2.5 und kann durchaus bei den jüngeren Versionen überboten werden.

Nachdem pppd für analoge Geräte doch recht dicht an den einzelnen Bytes und Bits konfiguriert werden muss, ist die Beschreibung der ISDN-Verbindung schon fast trivial, wenn auch umfangreicher. Zumindest ist sie wesentlich einfacher als es auf den ersten Blick erscheint. Das Prinzip ist dem des pppd ähnlich, doch kommt hier eine andere Reihenfolge zum Tragen: Die Verbindung wird nicht explizit durch einen vorgegebenen Dialog (mittels chat), sondern durch die "Intelligenz" der ISDN-Karte selbst aufgebaut. Das Gerät wird also zunächst programmiert und dann erst zum Verbindungsaufbau veranlasst, sodass der Kernel selbst nicht für den Kommunikationsablauf Sorge tragen muss. Eine ISDN-Karte verhält sich also gewissermaßen wie eine Netzwerkkarte, die vom System wie eine "Blackbox" behandelt wird.

4.2.1 Hardware

Die Hardware ist leider nicht beliebig auswählbar, es existiert jedoch eine Liste der unterstützten Karten z.B. unter

http://www.suse.de/Support/sdb/isdn.html.

Die Unterstützung der Hardware ist allerdings - wie so vieles unter Linux - einem ständigen Fortschritt unterworfen, sodass es sich durchaus lohnt, in den eigenen Kernel-Sourcen in der Datei "Documentation/isdn/README.HiSax" nachzusehen, welche der erhältlichen Karten mittlerweile unterstützt werden. Zu jeder Karte müssen noch die benötigten Ressourcen bekannt sein, was bei "normalen" ISA-Karten mittels Jumper oder Setup-Software festzulegen ist. PCI-Karten sind noch unkomplizierter: Ihre Ressourcen müssen gar nicht angegeben werden, da diese beim Booten festgelegt und vom Treiber erkannt werden. Problematisch wird es allerdings bei ISA-PnP-Karten. Diese müssen entweder mittels "isapnp" oder aber vom PC-BIOS festgelegt werden. Bei letzterer Variante können die Werte übernommen werden, die ein eventuell vorhandenen DOS-Treiber auf demselben Rechner erkennt.

Zusätzlich bleibt noch zu sagen, dass allgemein ein direkter S0-Anschluss einer TK-Anlage vorzuziehen ist, weil sich nicht alle Anlagen an die Handbuchbeschreibungen halten. Insbesondere wird das innerhalb der TK-Anlage benutzte Protokoll häufig mit "Euro-ISDN" (DSS1) angegeben, während tatsächlich das alte, nationale 1TR6 verwendet wird. Wenn also der Betrieb an einer TK-Anlage nicht zu umgehen ist, und Probleme entstehen, hilft oft die Umstellung auf das andere Protokoll.

4.2.2 Software

Im Kernel selbst ist die ISDN-Konfiguration unter "ISDN subsystem" zu finden. Dort sollte das Subsystem auf jeden Fall als Modul eingebunden werden, wie auch der SyncPPP-Treiber und die Unterstützung des benutzten ISDN-Protokolls:

```
<M> ISDN support
 [*] Support synchronous PPP (NEW)
...
[*] HiSax Support for EURO/DSS1 (NEW)
[*] HiSax Support for german 1TR6 (NEW)
```

Sofern der Chipsatz auf der Karte bekannt ist, sollte das entsprechende Modul innerhalb des Subsystems ebenfalls aktiviert werden. Da es davon bereits extrem viele gibt, kann an dieser Stelle keine Standard-Antwort getroffen werden. Sofern jedoch eine aktive ISDN-Karte vor-

liegt, sollte die CAPI-Unterstützung aktiviert werden, um evtl. weitere Möglichkeiten nutzen zu können

```
< > AVM-B1 with CAPI2.0 support (NEW)
```

Nach der Kompilation und Installation der Module und des Kernels sollte hisax.o, isdn_bsdcomp.o und isdn.o im Modules-Verzeichnis vorhanden sein, um die Unterstützung zu gewährleisten.

Außer den Kernel-Modulen wird das Paket "i4l" (ISDN for Linux) benötigt, das den Daemon "ipppd", einige andere Kontrollmodule und sehr viel Dokumentation enthält.

Der isdnlog protokolliert die ISDN-Calls, und ist eigentlich optional, doch weil zumindest die Fehlersuche durch ihn erleichtert wird, sollte er konfiguriert werden. Zentrale Datei dabei ist "/etc/isdn/isdn.conf". Diese wird bei der Installation von i4l automatisch eingerichtet, muss jedoch noch mit eigenen Ergänzungen komplettiert werden.

Die SuSE-Distribution liefert bereits eine an deutsche Verhältnisse angepasste Version in der folgende Stellen beachtet werden müssen:

```
[GLOBAL]
COUNTRYPREFIX   = +
COUNTRYCODE     = 49
AREAPREFIX      = 0

# EDIT THIS LINE:
AREACODE        = 911

# Example:
#AREACODE        = 911 # Nuernberg
Alias-Namen
```

Die einzige Stelle die noch ergänzt werden muss, ist der "AREACODE", also die Ortsvorwahl ohne die führende Null. Noch interessanter ist die Einrichtung von Alias-Namen bekannter Nummern. Sofern ein Alias für eine Nummer existiert, wird in der Protokolldatei nicht mehr die Nummer, sondern der Name beschrieben, was die Lesbarkeit deutlich erhöht. In der Datei "/etc/isdn/callerid.conf" können solche Nummern nach folgendem Schema ergänzt werden

```
[MSN]
NUMBER = +4912345678
ALIAS  = irgendwer
ZONE   = 1
```

Sind alle Optionen festgelegt, muss der isdnlog mit (mindestens) folgenden Parametern während des Bootvorgangs gestartet werden (aus dem README von Andreas Kool):

```
/sbin/isdnlog -nsS -v1 -w1 -m0x5f7 \ /dev/isdnctrl
```

Das Modul hisax stellt die eigentliche Treibersoftware für die ISDN-Karte dar und muss - sofern es sich nicht um eine PCI-Karte handelt - die verwendeten Ressourcen mittels Parameter bekommen. Der Aufruf sieht prinzipiell so aus:

```
modprobe hisax id=<idstring> \
io=<io-port> \
irq=<interrupt> \
mem=<memory-base> \
type=<Kartentyp> \
protocol=<Protokollnummer>
```

Der ID-String ist prinzipiell frei wählbar, er dient als Erkennungsmerkmal für den Kernel als Bindung zu einer Karte. In der Kernel-Dokumentation kann unter "Documentation/isdn/READ-

ME.HiSax" nachgelesen werden, welche Karte welchem Typ entspricht und welche Parameter sie erwartet:

```
 1   Teles 16.0                irq, mem, io
 2   Teles  8.0                irq, mem
 3   Teles 16.3 (non PnP)      irq, io
 4   Creatix/Teles PnP         irq, io0 (ISAC), io1 (HSCX)
 5   AVM A1 (Fritz)            irq, io
 6   ELSA PCC/PCF cards        io or nothing for autodetect
(the iobase is required only if you have more than one ELSA
card in your PC)
 7   ELSA Quickstep 1000       irq, io  (from isapnp setup)
 8   Teles 16.3 PCMCIA         irq, io
 9   ITK ix1-micro Rev.2       irq, io
10   ELSA PCMCIA               irq, io  (set with card manager)
11   Eicon.Diehl Diva ISA PnP  irq, io
11   Eicon.Diehl Diva PCI      no parameter
12   ASUS COM ISDNLink         irq, io  (from isapnp setup)
13   HFC-2BS0 based cards      irq, io
14   Teles 16.3c PnP           irq, io
15   Sedlbauer Speed Card      irq, io
15   Sedlbauer PC/104          irq, io
15   Sedlbauer Speed PCI       no parameter
16   USR Sportster internal    irq, io
17   MIC card                  irq, io
18   ELSA Quickstep 1000PCI    no parameter
19   Compaq ISDN S0 ISA card   irq, io0, io1, io (from isapnp
setup io=IO2)
20   NETjet PCI card           no parameter
21   Teles PCI                 no parameter
22   Sedlbauer Speed Star (PCMCIA) irq, io (set with card ma-
nager)
24   Dr. Neuhaus Niccy PnP     irq, io0, io1 (from isapnp se-
tup)
24   Dr. Neuhaus Niccy PCI     no parameter
25   Teles S0Box               irq, io (of the used lpt port)
26   AVM A1 PCMCIA (Fritz!)    irq, io (set with card manager)
27   AVM PnP (Fritz!PnP)       irq, io  (from isapnp setup)
27   AVM PCI (Fritz!PCI)       no parameter
28   Sedlbauer Speed Fax+      irq, io (from isapnp setup)
```

Der Parameter "Protokoll" kann nur 3 Werte annehmen:

- 1 bezeichnet das deutsche Protokoll 1TR6
- 2 steht für das Euro-ISDN (EDSS1)
- 3 kann für Standleitungen eingesetzt werden, (leased lines)

Die Ergebnisse und Ereignisse des hisax-Treibers können in der allgemeinen Protokolldatei "/var/log/messages" mitverfolgt werden.

Im Gegensatz zu dem pppd, der das Netz-Interface erst nach seinem Start selbst erzeugt, erwartet das Gegenstück der ISDN-Verbindungen der "ipppd" bereits ein entsprechendes Gerät, das mittels "isdnctrl" angelegt wird. In diesem virtuellen Gerät werden alle Informationen für die Anwahl gespeichert, sodass der ipppd darauf zurückgreifen kann. Die Namen der ISDN-Geräte sind zwar prinzipiell frei definierbar, jedoch sollte aus technischen Gründen ein Name wie "ippp0" gewählt werden. Die Letzte Ziffer bezeichnet die Nummer der angesprochenen ISDN-Karte und stellt die Analogie zu den Netzwerkkarten (eth0, eth1 etc.) her. Ein Beispielskript sähe so aus:

```
# /etc/ppp/isdnctrl-up
```

```
#
# zunaechst das Geraet definieren
isdnctrl addif ippp0

# Eigene Nummer (MSN bzw. EAZ, wobei *immer*
# die Option mit "eaz" angegeben wird)
isdnctrl eaz ippp0 12

# Nummer, die gewaehlt wird
isdnctrl addphone ippp0 12345

# Nummern, die anrufen duerfen (erstmal alle)
isdnctrl addphone ippp0 in

# Sicherheit einschalten: nur explizit
# angegebene Nummern duerfen rein
isdnctrl secure ippp0 on

# Protokoll (Layer 2): eins aus
# (x75i, x75ui, x75bui, hdlc)
isdnctrl l2_prot ippp0 hdlc

# Protokoll (Layer 3)
isdnctrl l3_prot ippp0 trans

# Encapsulation: eins aus
# (rawip, cisco_h, syncppp)
isdnctrl encap ippp0 syncppp

# Sekunden Lethargie bis zum Auflegen
isdnctrl huptimeout ippp0 60

# max. Waehlversuche
isdnctrl dialmax ippp0 5

# PPP mit device verbinden
isdnctrl pppbind ippp0 0

# IP-Konfig. wie Netzkarte. Adressen sind
# egal, wenn beide bekannt sind, sollten sie
# hier eingesetzt werden.
ifconfig ippp0 192.168.8.1 pointtopoint \
192.168.8.100 up

# Erfolg = OPEN melden
isdnctrl verbose 3
Wenn dieses Gerät wieder gelöscht werden soll, geht das recht
einfach:
# /etc/ppp/isdnctrl-down
#
ifconfig ippp0 down
isdnctrl delif ippp0
```

Nun ist es endlich so weit, es kann der Verbindungs-Daemon gestartet werden. Dieser Dienst, der fast identisch wie der pppd zu benutzen ist, benutzt auch ähnliche Konfigurationsdateien:

- Die globale "/etc/ppp/ioptions", die für alle ISDN-Geräte zuständig ist (sofern mehr als eine vorhanden ist).
- Die Kommandozeile, in der alle Parameter "spaghettiartig" hintereinander geschrieben werden.

Als Alternative kann in der Kommandozeile mit dem Parameter "-file" eine selbstgewählte Konfigurationsdatei einbezogen werden. Bei nur einer Verbindung kann der Aufwand aber minimal bleiben und die globale Datei benutzt werden:

```
# /etc/ppp/ioptions
# globale Konfigurationsdatei fuer ISDN

# Zunaechst das Device wie mit isdnctrl
# angelegt:
/dev/ippp0

# vorgegebene IP-Adressen: <local>:<remote>
# wird hier vom Server bezogen
0.0.0.0:
# Username zum Anmelden
user meinname
# fuer CHAP muss ein Rechnername her
name pcname

# jegliche Kompression verhindern
-vj
-vjcomp
-ac
-pc
-bsdcomp

# max receve unit festlegen
mru 1524
# max transmit unit festlegen
mru 1500
```

Die Kommandozeile für den Aufruf des ipppd sieht somit recht übersichtlich aus:

```
ipppd pidfile /var/run/ipppd.ipppd0.pid &
```

Durch die Art der Verbindungsherstellung fällt eine Login/Passwort-Übertragung wie beim CHAT aus. Die beiden Alternativen sind dieselben wie beim pppd: PAP und CHAP. Sinnvollerweise werden dieselben Dateien für diese Identifikation verwendet: "/etc/ppp/pap-secrets" und "/etc/ppp/chap-secrets". Eine CHAP-Datei hätte nach dem obigen Beispiel, dem PC-Namen "pcname", dem Servernamen "svname" und einem Passwort "geheim" folgenden Aufbau:

```
# client    server      pw     iplist
pcname      svname      "geheim"    *
svname      pcname      "geheim"    *
```

Im Gegensatz zu dem pppd ist das "Dial-on Demand" in dem ipppd fest eingebaut. Jedoch besteht wie beim pppd die Möglichkeit, dass die eigene IP-Adresse nach dem Verbindungsaufbau noch einmal verändert wird. Aus diesem Grund sollte auch der Befehl

```
echo "7" > /proc/sys/net/ipv4/ip_dynaddr
```

Benutzt werden, um die automatische Anpassung zu gewährleisten.

Nun soll aber nicht für jede Verbindung ein solcher Wust von Befehlen abgesetzt werden, wesentlich eleganter ist eine Startup-Datei, die in das entsprechende Verzeichnis eingefügt werden kann:

```
case "$1" in
    start)
        echo -n "Starting ipppd: "
        modprobe hisax id=hisax type=13 protocol=2
        /etc/ppp/isdnctrl-up
        /usr/sbin/ipppd pidfile /var/run/ipppd.ipppd0.pid &
```

```
    ;;
    stop)
        echo -n "Shutting down ipppd: "
        kill `cat /var/run/ipppd.ipppd0.pid`
        /etc/ppp/isdnctrl-down
```

5 DNS

5.1 Zum Verständnis

Das „**Domain Name System**“ (DNS) lässt sich am ehesten am Beispiel des Internets erklären. Eine große, auf viele Rechner verteilte Datenbank bildet die Basis des DNS. Jeder ans lnternet angeschlossene DNS-Server ist ein Teil dieser Datenbank. Durch die Einteilung des Internets in Domains gibt es eine klare Hierarchie. Eine Domain ist der Name eines Rechnernetzwerkes. Es gibt verschiedene Arten von Domains:

Top-Level-Domains:

Der Domainname dieser Art besteht aus dem Ländercode, beispielsweise „de“, „at“, „it“, etc., oder einer festgelegten Abkürzung für einen bestimmten Bereich, beispielsweise, „org“ für nichtkommerzielle Organisationen, “com“ für kommerzielle Organisationen oder „net“ für Netzwerkorganisationen.

Second-Level-Domains:

Eine „second level domain“ ist ein beigewählter Name, wie zum Beispiel die Domain der VHS Lippe „vhs-lippe“. Der Anfang der Domain gibt den Rechnernamen an, der vom Administrator bei der Installation des Betriebsystems vergeben wurde.

Wie nun aus diesen Komponenten ein Domainname, der sogenannte „**Fully Qualified Domain Name**“ (FQDN), gebildet wird, lässt sich anhand eines Beispiels verdeutlichen. Ziel ist es, den WWW-Server der VHS Lippe im Internet zu präsentieren. Daraus ergäbe sich folgende Zusammensetzung:

Die Top-Level-Domain würde „de“ bilden, weil Lippe sich in Deutschland befindet. Als Second-Level-Domain bietet sich „vhs-lippe“ an, schließlich ist der WWW-Server allein für die VHS Lippe zuständig. Ein sinnvoller Rechnername ist „www“, um die Art der angebotenen Dienste zu verdeutlichen. Der FQDN wäre folglich „www.vhs-lippe.de“. Der WWW-Server ist jetzt eindeutig im Internet identifizierbar.

Wie wird jedoch die IP-Adresse von „www.vhs-lippe.de“, die für die Netzkommunikation benötigt wird, auf diesen Namen beziehungsweise wie wird dieser Name auf die zugehörige IP-Adresse abgebildet? Jedem mit dem Internet verbundenen System ist mindestens ein **Nameserver** bekannt. Wenn also ein Request vom Computer „blah.irgendwo.de“ an den Rechner „www.vhs-lippe.de“ gesendet werden soll, wird der FQDN „www.vhs-lippe.de“ an den Nameserver von „blah“ übermittelt, mit der Aufgabe, die zugehörige IP-Adresse zu finden. Dieser Nameserver startet nun einen Ablauf, der „Namensauflösung“ genannt wird. Zuerst fragt er bei einem der Root-Nameserver, welcher DNS-Server für die Namensauflösung der Top-Level-Domain „de“ zuständig ist. Nachdem er die benötigten Informationen erhalten hat, wendet er sich an den Computer, der die DNS-Funktionen für „de“ bereitstellt. Von dem wird ihm der Nameserver der Domain „vhs-lippe“ genannt, dem er daraufhin den Rechnernamen „www“ zur Namensauflösung schickt. Sobald er das Ergebnis der Namensauflösung empfangen hat, wird es an den Rechner „blah“ zurückgeschickt, der nun in der Lage ist, einen Request an „www.vhs-lippe.de“ zu senden.

Vermutlich werden Sie sich jetzt fragen, warum das System so umständlich ist, schließlich hätte man doch auch einen eindeutigen Namen zuordnen können, ohne Top-Level- und Second-Level-Domains. Tatsächlich wurde dies ursprünglich so gehandhabt. Sollte ein Rechner ins Internet eingegliedert werden, musste sich die verantwortliche Person beim NIC eintragen und die Datei „HOSTS.TXT“ herunterladen, um diese dann als „/etc/hosts“ zu verwenden. Solange das

Internet noch klein und überschaubar war, stellte dieses System eine einfache Möglichkeit der Namensauflösung dar. Doch das Internet wuchs in den letzten Jahren derartig an, dass die Einführung einer professionellen Namensauflösung notwendig wurde. Die jetzt folgende Beschreibung ist nicht vollständig, reicht aber für den Gebrauch in lokalen Netzen mit einem Internet-Router aus.

5.2 DNS und BIND

BIND ist ein Softwarepaket, das die Funktionen des DNS bereitstellt. BIND gibt es in der Version 4.9.7 und 8.2.x. Da die 4er-Version veraltet ist und Sicherheitsmängel aufweist, sei hier lediglich die Version 8.x beschrieben. BIND darf unter der Beachtung der GPL kostenlos benutzt werden. Sollte BIND nicht standardmäßig als Paket mit ihrer Linux-Distribution mitgeliefert worden sein, kann das Paket von „ftp.isc.org/isc/bind/src/cur/bind8-src.tar.gz“ bezogen werden.

5.2.1 Konfiguration von BIND

Bei der Einrichtung von BIND wird an dieser Stelle davon ausgegangen, dass es sich um ein lokales Netz handelt, das nicht von außen erreicht werden muss oder darf. Anhand eines Beispiels lässt sich die Konfiguration am besten beschreiben. In einer Domain „planet.de“ gibt es fünf verschiedene Rechner mit folgenden Namen und dazugehörigen IP-Adressen:

```
merkur.planet.de          192.168.1.1
venus.planet.de           192.168.1.2
erde.planet.de            192.168.1.3
mars.planet.de            192.168.1.4
jupiter.planet.de         192.168.1.5
```

Von diesen Rechnern ist „erde“ auserwählt worden, die DNS-Dienste bereitzustellen

BIND benötigt mehrere Konfigurationsdateien, für die folgende Namen vergeben werden:

/etc/named.conf: Übermittelt BIND grundsätzliche Informationen über die Domain und Lage der anderen Dateien.

planet.zone: Diese Datei bildet die Hostnamen auf Adressen ab.

planet.rev: Hier werden die Adressen auf die Hostnamen abgebildet.

named.root: Enthält Informationen zu den aktuellen Root-Nameservern.

localhost.zone: Datei für die Abbildung der Localhost-Adresse (127.0.0.1) auf den jeweiligen Hostnamen

Der Inhalt der Datei planet.zone wird in dem folgenden Listing dargestellt.

```
;
;           /var/named/planet.zone
;     $ORIGIN planet.de
;
@     IN SOA            planet.de. root.erde.planet.de (
            2000020101  ;serial (yyyymmddrr)
            10800       ;Refresh
            46000       ;Retry
            568000      ;Expire
            86400 )     ;Minimum
            IN NS       erde.planet.de.
            IN NS       venus.planet.de.
planet.de.  IN MX 5     mail.planet.de.
```

```
;
www         IN CNAME    erde.planet.de.
;
merkur            IN A        192.168.10.1
venus             IN A        192.168.10.2
erde              IN A        192.168.10.3
mars              IN A        192.168.10.4
jupiter           IN A        192.168.10.2
```

Der erste Eintrag ist ein SOA-Resource-Record. Dieser Eintrag gibt an, dass er die beste Informationsquelle für die Zone „planet de“ im Bezug auf die Namensauflösung ist.

Vor dem DNS-Record wird dem DNS mitgeteilt, dass es sich bei den Daten um Daten der Klasse „Internet“ handelt, Dieser Eintrag ist optional und kann weggelassen werden. Deswegen wird hier auch nicht näher darauf eingegangen

Rechts neben dem SOA-Eintrag steht der primäre Nameserver für die Zone, in diesem Fall also erde.planet.de. Der nächste Eintrag ist eine Email-Adresse, wobei der erste Punkt durch ein »@« ersetzt werden muss. Die Email-Adresse ist die Adresse, unter welcher der Zonenadmin zu erreichen ist.

Danach folgen mehrere Einträge, die von BIND als ein einziger Eintrag gelesen werden, da alles in Klammern zusammengefasst ist. Diese Werte werden eigentlich nur von Slave-Nameservern verwendet und geben Zeiträume an, in denen Teile dieser Datenbank erneuert und aufgefrischt werden muss.

Der nächste Eintrag teilt BIND mit, welche Rechner im Netzwerk die Nameserver sind. Als Klasse wird wieder IN und als Ressource Record „Nameserver“ verwendet.

Im darauffolgenden Block wird die Abbildung der Namen auf Adressen eingetragen. Auch hier wird die Klasse IN verwendet, allerdings diesmal in Kombination mit dem Ressource-Rekord A für „address“.

Die Datei planet.rev ist im folgenden Listing aufgezeichnet. Am Anfang der Datei sieht wieder ein SOA-Record. Der SOA-Eintrag ist dem aus der Datei „planet.zone“ gleich, mit dem kleinem Unterschied, dass an erster Stelle der Datei nicht „planet.de“. steht, sondern 10.168.192.in-addr.arpa“. Im anderen Teil der Datei werden BIND zuerst die Nameserver mitgeteilt und dann eine Abbildung der Adressen auf Hostnamen definiert Der Ressource-Rekord „PTR“ bedeutet Pointer-Record und muss für eine „Rückwärtsauflösung“ verwendet werden.

```
;
;           /var/named/planet.rev
;
@     IN SOA            planet.de. root.erde.planet.de (
            2000020101  ;serial (yyyymmddrr)
            10800       ;Refresh
            46000       ;Retry
            568000      ;Expire
            86400)      ;Minimum
      IN NS       erde. planet.de.
;
;     $ORIGIN 10.168.192.in-addr.arpa.
;
;
localhost   IN A        127.0.0.1
;
```

```
1           IN PTR          merkur.planet.de.
2           IN PTR          venus.planet.de.
3           IN PTR          erde.planet.de.
4           IN PTR          mars.planet.de.
4           IN PTR          jupiter.planet.de.
;
```

Es kommt häufig vor, dass ein Host Daten an sich selber über die Loopback-Schnittstelle sendet. In den meisten Fällen ist die Netzwerkadresse des Loopback-Netzwerkes 127.0.0 und die IP-Adresse der Schnittstelle 127.0.0.1. Es ist zwar nicht nötig, BIND mitzuteilen, dass er bei der eben genannten Adresse den Namen „localhost“ beziehungsweise auf den Namen „localhost“ die IP-Adresse 127.0.01 zurückgeben soll, aber ohne diese Definition kann es zu bösen Überraschungen kommen.

Deshalb wird in der Datei „localhost.zone“ eine Namensauflösung für diese Schnittstelle beschrieben. Der Inhalt dieser Datei ist in dem folgenden Listing dargestellt.

```
;
;           /var/named/localhost.rev
;
@     IN SOA            obergoeker.de. root.gandalf.obergoe-
ker.de (
            2000020101  ;serial (yyyymmddrr)
            10800       ;Refresh
            46000       ;Retry
            568000      ;Expire
            86400 )     ;Minimum
      IN NS       gandalf.obergoeker.de.
;
1     IN PTR            localhost
```

Die nächste Konfigurationsdatei muss vom FTP-Server ftp.rs.internic.net (IP: 198.41.0.7) per „anonymous FTP“ heruntergeladen werden Der Name dieser Datei ist **named.root** und sie befindet sich auf dem Server im Unterverzeichnis „domain“. Diese Datei beinhaltet die IP-Adressen der „root-Nameserver“ des Internets.

Zum Abschluss gilt es die Datei „/etc/named.conf“ zu bearbeiten. Hier wird BIND mitgeteilt, wo die anderen Konfigurationsdateien zu finden sind. Die vorhandene Datei kann prinzipiell so, wie sie ist übernommen werden, lediglich die „zone“-Einträge und die Pfadnamen müssen abgeglichen werden. Um nicht alles abdrucken zu müssen und um eine Übersicht zu gewährleisten, sind hier nur die zu verändernden Parameter abgebildet:

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

Zone "planet.de" IN {
      type master;
      file "planet.zone";
};

Zone "10.168.192.in-addr.arpa" IN {
      type master;
      file "planet.rev";
};
```

```
Zone "0.0.127.in-addr.arpa" IN {
      type master;
      file "localhost.rev";
};

Zone "." IN {
      type hint;
      file "named.root";
};
```

5.2.2 DNS-Server starten

Zum start des DNS-Servers muss lediglich das Kommando „/etc/named“ ausgeführt werden. Für den Fall, dass die Konfigurationsdatei nicht „/etc/named.conf“ ist, wird der Befehl „/etc/named -b <*Konfigurationsdatei*> eingegeben. Sofern beim Start Fehler aufgetreten sind, werden sie in der syslog-Datei „/var/log/messages“ mitprotokolliert. Deshalb sollte diese auf jeden Fall untersucht werden. Wenn Sie mit syslogd nicht vertraut sind, dann tippen Sie auf der Kommandozeile „man syslogd“ ein, um mehr Informationen über diesen Daemon zu erhalten.

Wenn der start laut syslogd fehlerfrei verlaufen ist, dann kann die Konfiguration des Nameservers, mit „nslookup“ überprüft werden. Dieser Befehl ist ein Tool, mit dem jede Art von Ressource-Rekords überprüft werden kann. So sollte nach der obigen Konfiguration der Befehl (auf „erde“ ausgeführt)

```
nslookup merkur.planet.de
```

die folgende Ausgabe liefern:

```
Server:  erde.planet.de
Address: 192.168.10.3

Name:    merkur.planet.de
Address: 192.168.10.1
```

Wenn der umgekehrte Fall, also

```
nslookup 192.168.10.1
```

ebenfalls das richtige Ergebnis (hier: „merkur.planet.de) liefert, ist die Konfiguration korrekt.

6 Samba

6.1 Entstehung

Samba wurde entwickelt, um einen Ausweg aus dem Dilemma zu finden, das durch die Verschiedenheit der Unix- und Microsoftwelt entstand. Unter Unix wurden und werden das Protokoll TCP/IP und für Fileserverdienste NFS eingesetzt. In der Microsoftwelt herrschte jedoch das Protokoll NetBIOS vor, während für Filserverdienste SMB (Server Message Blocks) benutzt wird. Um beide Welten zu verbinden musste auf den Microsoft-PCs üblicherweise recht teure NFS-Client-Software eingesetzt werden.

Seitdem nun auch Microsoft das TCP/IP nicht mehr ignorieren konnte, wurde das SMB dahingehend erweitert, dass es auch TCP/IP als Grundlage verwenden kann. Unter dieser Prämisse konnten einige Entwickler in Australien das SMB für Unix-Systeme entwickeln (Re-Engineering). Samba ist daher keinesfalls eine Linux-spezifische Software, sie wurde lediglich (unter anderem) an Linux angepasst.

Diese Software untersteht der GNU GPL, ist also im Quellcode verfügbar und somit an jede Plattform anpassbar. In der Tat ist Samba bereits an alle gängigen Unix-Plattformen angepasst worden.

6.2 Einsatzbereiche von Samba

Sinn und Zweck von Samba ist es, die betreffende Unix-Maschine in ein reines Windows NT/9x-Netzwerk zu integrieren, ohne auf den Windows-Maschinen zusätzliche Software installieren zu müssen.

Der Samba-Server stellt dabei Verzeichnisse in einem beliebigen Filesystem als die von Windows-bekannten „Shares“ und die eigenen Drucker zur Verfügung. Die Zugriffsrechte werden dabei vom Betriebssystem selbst bestimmt, so dass eine Transparenz sowohl von der Unix- als auch von de Windows-Seite aus gegeben ist. Der Server selbst kann als dediziertes System mit einer eigenen Benutzerverwaltung betrieben oder aber in eine NT-Domäne integriert werden. Letzteres ist besonders dann sinnvoll, wenn bereits eine NT-Domäne besteht.

6.3 Schnellinstallation

Dank der guten Unterstützung der Linux-Distributoren ist Samba als Softwarepaket fast jeder Distribution beigelegt, sodass es nur mit der jeweiligen Verwaltungskomponente dem System hinzugefügt werden muss. Aber auch in Form der Sourcen ist die Installation mittlerweile recht einfach: Es reicht der klassische „Gnu-Dreisprung“ der folgenden Befehle:

```
./configure
make
make install
```

Der letzte Befehl kopiert die beiden Binärdateien „smbd“ und „nmbd“ in das Verzeichnis /usr/local/bin womit sie in den meisten Distributionen direkt ausführbar sind. Ist diese Position nicht korrekt, müssen sie im einfachsten Fall manuell kopiert werden.

Der wichtigste Punkt ist jedoch die Datei „smb.conf“, die als zentrale Konfigurationsdatei auf jeden Fall angepasst werden muss. Die Installation stellt dabei eine Demo-Datei zur Verfügung, die Minimalansprüchen bereits genügt und die sich im Verzeichnis „/etc“ oder „/etc/samba“ befindet:

```
[global]
```

```
   workgroup = arbeitsgruppe
   guest account = nobody
   keep alive = 30
   os level = 2
   security = user

   printing = bsd
   printcap name = /etc/printcap
   load printers = yes

   socket options = TCP_NODELAY
   map to guest = Bad User
   wins support = no

[homes]
   comment = Heimatverzeichnis
   browseable = no
   read only = no
   create mode = 0750

[printers]
   comment = All Printers
   browseable = no
   printable = yes
   public = no
   read only = yes
   create mode = 0700
   directory = /tmp
```

Diese Konfiguration reicht bereits aus, um eines ersten Test zu versuchen. Lediglich der Name der „Workgroup“ sollte vorher an die verwendete „Arbeitsgruppe“ der Windows-Rechner angepasst werden.

Wird der Samba-Server gestartet, stehen bereits die Drucker und die Home-Directories der User zur Verfügung. Voraussetzung ist lediglich, dass der betroffene User unter Windows mit derselben Kennung/Passwort-Kombination angemeldet ist, die unter Linux erwartet wird. Soll ein Verzeichnis freigegeben werden, dass für alle bekannten User gleichermaßen verfügbar ist, so reicht ein solcher Eintrag:

```
[daten]
   comment = "Allg. Daten"
   path = /daten
   browseable = yes
   read only = no
   create mode = 0750
```

In diesem Fall wird das Verzeichnis „/daten“ unter dem Namen „daten“ freigegeben. Heißt der Serverrechner beispielsweise „gandalf“, so lautet der Windows-Befehl zu Einbinden dieses „Shares“ auf Laufwerk „X:“:

```
net use x: \\gandalf\daten
```

Alternativ kann die Anbindung natürlich auch interaktiv über den Explorer vorgenommen werden.

Ein Windows-NT/2000/XP-Rechner meldet mit dieser Konfiguration ein ungültigen Zugang, weil Samba die Passwörter im Klartext erwartet und nicht – wie seit NT SP3 es in der Windows-Welt üblich ist – in verschlüsselter Form. Da sich diese Art der Passwortverschlüsselung nicht ohne signifikante Änderungen im Betriebssystem-Kern in ein UNIX-System integrieren lässt, muss eine separate Datei „*smbpasswd*“ angelegt werden, die die neuen Passwörter aufnimmt. Mit dem Befehl

```
smbpasswd -a <benutzer>
```

wird der betreffende Benutzer in dieser Datei ergänzt und das Passwort gesetzt. Die Option „-a“ ist dabei nur für den ersten Aufruf dieses Kontos notwendig, zur Passwortänderung muss sie weggelassen werden.

6.4 Samba in der NT-Domäne

6.4.1 NT-Domänen

Eine besondere Anforderung an Samba ist nach wie vor die Integration in eine bestehende NT-Welt. Ein solcher Zusammenschluss on NT-Systemen wird von Microsoft selbst als "Domäne" (oder im englischen Sprachgebrauch "domain") bezeichnet, die unabhängig von den Internet-Domains existiert. Eine solche Domäne zeichnet sich dadurch aus, dass die Benutzer und deren Rechte nicht mehr auf den einzelnen PCs administriert werden, sondern auf einem einzigen Server, der alle anderen Rechner, die sich dieser Domäne angeschlossen haben, synchronisiert. Das Prinzip der NT-Domänen entspricht weitestgehend dem des NIS unter UNIX-Systemen.

Dieser "primäre" Server, der als "**Primary Domain Controller**" (=PDC) bezeichnet wird, muss genau einmal in jeder Domäne existieren. Er kann zwar von weiteren Servern unterstützt werden, die für die Client-Rechner dieselben Aufgaben übernehmen können, es darf aber immer nur ein Rechner die "Oberherrschaft" besitzen. Die unterstützenden Server werden als "**Backup Domain Controller**" (=BDC) bezeichnet, müssen aber nicht notwendigerweise existieren. Meldet ein Benutzer sich an einem Arbeitsplatzrechner an, so überprüft dieser mit Hilfe eines Domain Controllers die Benutzer/Passwort-Kombination. Welcher Controller die Zulässigkeit bestätigt, hängt davon ab, wie schnell die Server auf die Anfrage reagieren (der Erste gewinnt). Soll aber eine Änderung an einem Benutzerkonto erfolgen (andere Rechte, anderes Passwort etc.), so muss zwingend der PDC vorhanden sein, um die Master-Datenbank der Benutzer verändern zu können.

Jeder Rechner – egal ob Server oder Arbeitsstation – muss vom NT-Superuser, dem "Administrator" der Domäne in dieser angemeldet werden. Das Prinzip ist dem der Benutzerkonten recht ähnlich und es sogt dafür, dass nur Rechner mit den Domänen-Benutzerkonten versorgt werden, die dem Administrator auch bekannt, "vertrauenswürdig" sind.

Fällt der PDC aus, kann ein BDC dessen Position übernehmen, wobei dieser explizit vom Administrator zum PDC "erhoben" werden muss. Wird der ehemalige PDC wieder in die Domäne integriert, muss er zwingenderweise zunächst zum BDC werden, bevor der ursprüngliche Zustand wieder manuell hergestellt werden kann.

6.4.2 Position des Linux-Servers

Der Linux-Server muss in der NT-Domäne eine Zwischenposition einnehmen. Aus Sicht der Server verhält er sich wie eine Workstation, aus Sicht der Workstations ist er ein Server. Die Anmeldung der Workstations an den Linux-Server geschieht zweischichtig. Im ersten Schritt überprüft samba, ob der User mit dem angegebenen Passwort an einem Windows-Domain-Controller angemeldet werden kann. Im zweiten Schritt überprüft Samba, ob der Benutzername im lokalen System existiert und das Passwort mit der smbpasswd validiert werden kann. Erhält Samba in beiden Fällen eine positive Antwort, wird der Share an den User freigegeben. Ist jedoch auch nur eine negativ, so wird die Anmeldung verweigert.

Es müssen also zwingenderweise alle NT-Benutzer, die sich am Linux-Server anmelden sollen, auch in derselben Schreibweise als Benutzer im Linux-System befinden, so dass eine doppelte Benutzerverwaltung notwendig wird. Darüber hinaus muss der Linux-Rechner, der den Samba-

Server beherbergt, ebenfalls in der NT-Domäne als gültiger Server eingetragen sein, da sich aus Sicht des PDCs bei der Passwortprüfung der Benutzer von dem Linux-Rechner aus anmeldet.

6.4.3 Anpassung der smb.conf

Die Anpassung von der Samba-Seite findet ausschließlich in der "globals"- Sektion der "smb.-conf" statt. Um das obige Beispiel abzubilden, sollte sie dabei folgendermaßen aussehen:

```
workgroup = ntgruppe
```

Der Name der Workgroup muss zwingend mit dem Domänennamen übereinstimmen.

```
guest account = nobody
```

Der Gast-Account wird zum "browsen" benutzt. Ist er keinem gültigen Linux-User zugeordnet, bleibt dieser Rechner für die Arbeitsstationen "unsichtbar". Aus diesem Grund muss auch überprüft werden, ob der hier angegebene Linux-Account auch existiert.

```
security = DOMAIN
password server = primus
```

Der Eintrag "security = **domain**" arbeitet ähnlich wie der Wert "server", er erzwingt also die Identifikation bei dem nachfolgend genannten Server, doch ermöglicht der Eintrag "**domain**" eine bessere Umsetzung des Windows-eigenen Konzeptes. daher sollte in NT/Win2000-Domänen nur "domain" verwendet werden. Der Eintrag „passwort server“ gibt den Rechner an, bei dem die Benutzerkonten validiert werden. Der angegebene Name wird nach DNS bzw. NetBEUI aufgelöst und muss einen der Domain Controller darstellen. In einem lokalen Subnetz kann Samba den PDC durch einen Broadcast nach LanManager-Methode ermitteln. In diesem Fall reicht der Wert „*“.

```
encrypt passwords = Yes
```

Da sich Windows NT defaultmäßig weigert, Passwörter in unverschlüsselter Form zu übermitteln, müssen diese unter Samba explizit eingeschaltet werden. Der security- Eintrag sorgt dafür, dass die Passwörter zunächst beim Passwort-Server abgefragt werden. Sollte diese Überprüfung negativ sein oder ist der Server nicht vorhanden, wird die lokale Datei "*smbpasswd*" konsultiert. In diesem Fall muss jeder Linux-User, der sich auch unter Samba anmelden soll, explizit mit dem Befehl "smbpasswd –a <user>" dort angelegt werden. In beiden Fällen muss jedoch der Benutzer als lokaler Linux-User in der "*passwd*" auffindbar sein.

```
interfaces = 192.168.2.1/24
```

Wenn die Maske (hier: /24) nicht mit der des NT-Netzwerkes übereinstimmt, kann sich der Samba-Server nicht mit Broadcasts bekannt geben und erscheint nicht in der Netzwerkliste der NT-Rechner

```
announce version = 2.0
```

Mit diesem Wert wird vorgegeben, welche NT-Serverversion der Samba-Server vorgibt zu sein. Der Defaulteintrag ist seit Samba 2.0 die Version "4.2", was dazu führt, dass sich der samba-Server bei fehlendem PDC automatisch zum Master-Browser erhebt, weil er die höchste Versionsnummer von allen Servern besitzt. Wird der Original-PDC wieder in das Netz integriert, weigert dieser sich seinen Browser-Dienst aufzunehmen, bis Samba wieder deaktiviert wird.

```
wins support = no
wins server = 192.168.2.10
```

Sofern WINS (Windows-Nameservice) benutzt wird, sollte Samba so eingestellt sein, dass nicht er selbst, sondern ein NT-Server (meistens der PDC) den WINS-Dienst übernimmt.

```
preferred master = no
local master = no
```

```
domain master = no
```

Diese drei Zeilen setzen fest, dass sich Samba nie zum "Master Browser" erhebt, was unter Umständen Komplikationen mit bereits vorhandenen PDCs oder BDCs verursacht

Hier noch einmal eine komplette "globals"-Sektion, wie sie in einer NT-Domäne benutzt werden kann.

```
workgroup = ntgruppe
netbios name = gandalf
server string = Linux-Server Gandalf
guest account = nobody
keep alive = 30
os level = 2
security = DOMAIN
password server = primus
encrypt passwords = Yes
printing = bsd
printcap name = /etc/printcap
load printers = yes
interfaces = 192.168.2.1/24
wins support = no
announce version = 2.0
wins server = 192.168.2.10
preferred master = no
local master = no
domain master = no
```

Die nachfolgenden Sektionen "Shares" und "Password" bleiben unverändert.

6.4.4 Samba-Integration

Soll nun der Linux-Rechner in die NT-Domäne integriert werden, müssen alle NT-Maschinen über TCP/IP kommunizieren können, d.h. Alle Server und die beteiligten Arbeitsplatzrechner müssen das TCP/IP als Protokoll in ihrem Netzwerksystem installiert haben. Samba beherrscht nicht die Kommunikation über NetBIOS oder SPX/IPX, sodass diese Protokolle nicht zur Auswahl stehen.

6.4.4.1 hosts und lmhosts

Der Linux-Server muss in das DNS der NT-Domäne aufgenommen werden, oder aber in der "hosts" und "lmhosts"-Dateien aller Rechner auftauchen, um ihn von einem anderen Rechner ansprechen zu können. Die Position dieser Dateien ist je System sehr unterschiedlich, wie die folgende Tabelle zeigt:

System	(übliche) Position von hosts/lmhosts
Linux	/etc
Windows 9x	C:\Windows
Windows NT/2000/XP	C:\winnt\system32\drivers\etc

Als nächsten Schritt muss der Linux-Rechner in der NT-Domäne bekannt gegeben werden. Dieses erfordert wiederum 2 Aktionen: Zuerst muss der Linux-Rechner mit seinem Namen in der NT-Domäne als "Server oder Workstation" im NT-Servermanager bzw. AD bekannt gegeben werden. Anschließend muss der Linux-Rechner eine NT-SID (=System-ID) vorweisen und beim PDC registrieren lassen. Nach dem obigen Beispiel (die Domäne heißt "ntgruppe") geschieht das (unter Linux) mit dem folgenden Befehl:

```
smbpasswd -j ntgruppe -r primus
```

Erscheint daraufhin die folgende Meldung, denn war die Registrierung erfolgreich:

```
smbpasswd: Joined domain ntgruppe
```

6.4.4.2 Benutzerintegration

Der zweite und aufwändigere Schritt ist die Integration der NT-User in das Samba-Konzept. Zu diesem Zweck muss jeder User erst in der NT-Domäne, dann unter Linux und zuletzt im Samba-System angelegt werden. Der letzte Schritt würde entfallen, wenn mit unverschlüsselten Passwörtern gearbeitet wird. Domänen-Konfigurationen können jedoch nur mit verschlüsselten Passwörtern arbeiten, so dass der folgende Eintrag in der "*smb.conf*" zu finden sein sollte:

```
encrypt passwords = Yes
```

Nachdem der User in der NT-Domäne bekannt ist, muss er zunächst im Linux-System eingerichtet werden. Je nach Distribution geschieht das mit dem jeweiligen Werkzeug (linuxconf, yast etc.) oder manuell mit diesem Befehl:

```
useradd –g <group> <name>
```

Während Windows beim Benutzernamen nicht zwischen Groß- und Kleinschreibung unterscheidet, ist es für Linux essentiell, dass die Namen in beiden Betriebssystemen identisch geschrieben werden.

In dieser Konfiguration werden die Passwörter im Normalfall am angegebenen Passwort-Server validiert, so dass der Benutzer subjektiv nur ein einiges Passwort für alles Systeme benötigt. Ein zusätzlicher Passworteintrag in der "*smbpasswd*" ist nur dann erforderlich, wenn der DC nicht verfügbar ist und trotzdem ein Login möglich sein soll.

6.4.4.3 Automatische Benutzerübernahme

Die Benutzerkonten an mehreren Stellen zu pflegen ist nicht nur umständlich, sondern auch fehlerträchtig. Sinnvoller ist es daher, die Synchronisation beider Systeme zu automatisieren, was durch die Variablen „*add user script*“ und „*delete user script*“ eingerichtet werden kann. Voraussetzung ist dabei, dass die Variable „*security*“ den Wert „domain“ oder „server“ enthält und der Passwort Server korrekt gesetzt und verfügbar ist. Versucht nun, ein Benutzer eine Verbindung zum Linux-System herzustellen, der bislang nicht in der „passwd“ enthalten war, prüft der Samba-Server zunächst die Gültigkeit des Zuganges am eingetragenen Passwort-Server. Ist dieser korrekt, versucht Samba den Benutzer lokal zu autorisieren. Ist dieser nicht in der „*passwd*“ enthalten, wird der Befehl ausgeführt, der unter „*add user script*“ eingetragen wurde. Erst wenn anschließend immer noch keine Benutzerkennung vorhanden ist, wird die Verbindung abgewiesen. Eine besondere Bedeutung kommt dabei dem Platzhalter „%u“ zu, da dieser zur Laufzeit mit dem eingetragenen Benutzernamen ersetzt wird. Ein möglicher und funktionierender Eintrag wäre der folgende:

```
add user script = useradd –g ntuser –s /bin/false %u
```

Bei Problemen ist es hilfreich, den angegebenen Befehl manuell von der Shell aus zu starten, um Syntax- und Konfigurationsfehler auszuschließen.

Analog gibt es auch die Möglichkeit, die in der Windows-Domäne bereits entfernten Konten auch auf dem Linux-System zu löschen. Dazu wird die Variable „*delete user script*“ verwendet, die nach einer fehlgeschlagenen Autorisierung am Passwort-Server den lokalen Benutzer ebenfalls löschen kann:

```
delete user script = userdel %u
```

Leider funktioniert diese Form nur in einer Richtung, indem Samba die lokalen Konten mit den Windows-Konten abgleicht, nicht umgekehrt.

Hier noch einmal eine komplette "globals"-Sektion, mit den beschriebenen Änderungen:

```
workgroup = ntgruppe
netbios name = gandalf
server string = Linux-Server Gandalf
guest account = nobody
keep alive = 30
os level = 2
security = DOMAIN
password server = primus
encrypt passwords = Yes
add user script = useradd -g ntuser -s /bin/false %u
delete user script = userdel %u
printing = bsd
printcap name = /etc/printcap
load printers = yes
interfaces = 192.168.2.1/24
wins support = no
announce version = 2.0
wins server = 192.168.2.10
preferred master = no
local master = no
domain master = no
```

6.4.5 Active Directory und „winbind“

Die eleganteste Methode für die Benutzerübernahme ist jedoch die Verbindung mit dem Tool „*winbindd*“. Wird Samba als „Mitgliedsserver“ eingesetzt, so können mit diesem Dienst die Benutzerinformationen direkt und nicht über den Umweg über die *passwd*-Einträge durchgeführt werden. Dazu wird ein Teil des Linux-Systems verwendet, der als „Pluggable Authentication Module“ bezeichnet wird. Über dieses Modul können beliebige Software-Bausteine in das Autorisierungs- und Identifizierungssystem eingebaut werden. Das Ursprüngliche Unix kannte nur die Dateien „*passwd*“ und vielleicht noch „*shadow*“, die für die Benutzerverwaltung verwendet wurden, andere Autorisierungssysteme (im Beispiel von NIS) mussten nachträglich in den Kernel hineingebaut werden. Da im Laufe des technischen Fortschrittes weitere Verfahren hinzukamen (LDAP, Datenbanken etc.) wurde das PAM entwickelt, das im Laufenden Betrieb umgebaut werden kann.

Weiterhin gibt es noch den globalen Name Service, der alle Dienste zusammenfasst, die zu irgendwelchen Objekten des Betriebssystems sprechende Namen liefern. Dazu gehört somit nicht nur der bereits erwähnte DNS, sondern auch die Zuweisung von Benutzernamen, Benutzergruppen. Auch diese Dienste können über Module in ihrer Funktionalität erweitert werden, indem vorgegebene Programmteile zur Laufzeit dynamisch vom Betriebssystem geladen und benutzt werden (shared Libraries).

Der erste Schritt besteht darin, die Module für die Namensauflösung in eine Reihenfolge zu bringen. Zu diesem Zweck wird die Datei „*./etc/nsswitch.conf*“ so geändert, dass sie die Dienste „passwd“ und „login“ zunächst auf die lokalen Dateien (files) lenkt und erst, wenn dort kein Eintrag gefunden werden kann, auf den „**winbind**“. Da die Authentifizierung in einer AD-Umgebung aber Kerberos-Tickets voraussetzt, muss zusätzlich ein **Kerberos-Client** installiert sein.

Um das gesamte Bild zu vervollständigen, kann zusätzlich das PAM-Mount Modul eingesetzt werden um Windows-Shares automatisch beim Einloggen des Benutzers zu mounten

Diese Komponenten **Samba, Winbind, Kerberos** und **PAM-Mount** müssen aufeinander abge-

stimmt und an die existierende AD-Umgebung angepasst werden, was zwangsläufig mehr Aufwand bedeutet als eine native Verbindung über NFS zu realisieren. Allerdings bleibt so die administrative Sicht unverändert, so dass die Windows-Umgebung nicht verändert werden muss. Inkonsistenzen der Benutzerverwaltung können auf diesem Weg nicht entstehen.

In der folgenden Beschreibung werden beispielhaft Konfigurationen abgebildet, die jeweils auf die reale Konfiguration angepasst werden müssen. Um die Systemabhängigen Teile besser erkennen zu können, wurden diese ***fett und kursiv*** hervorgehoben.

In diesem Beispiel werden diese Eigenschaften verwendet:

Master Domain Controller: **winserver1**

Domain Controller: **winserver2**

Domäne: **RAUM22**

AD-Domäne: **raum22.vhs-dt.de**

6.4.5.1 Anpassung des Samba-Servers

Der erste Schritt besteht darin, dem Samba-System mitzuteilen, wie die AD-Umgebung konfiguritert ist. Insbesondere ist der Eintrag für den „ADS-Realm“ und die „Workgroup“ wichtig. Befindet sich der Domain Conroller nicht imselben Subnetz, so muss dieser auch mit seiner Adresse oder seinem Namen angegeben werden, sonst genügt ein"*", um Samba automatisch suchen zu lassen.

Die Winbind-Einträge legen fest, in welchem Zahlenintervall die AD-Benutzer und –Gruppen abgebildet werden. Das Intervall muss daher groß genug sein, um alle Objekte (Benutzer, Gruppen, Computer) aufnehmen zu können. Sollte der Zahlenbereich jemals verändert werden, müssen auch die Berechtigungen im Home-Directory des Benutzers angepasst werden.

/etc/samba/smb.conf

```
security = ads
realm = raum22.vhs-dt.de
password server = winserver1.raum22.vhs-dt.de

workgroup = RAUM22

server string = %h workstation

encrypt passwords = true
passdb backend = smbpasswd guest

wins support = no

   os level = 0
   domain master = no
   local master = no
   preferred master = no

   name resolve order = lmhosts host wins bcast

   winbind uid = 10000-30000
   winbind gid = 10000-30000
   winbind use default domain = yes
   template homedir = /home/RAUM22/%U
   template shell = /bin/bash

; ISOLATIN1 with euro sign
```

```
unix charset = iso-8859-15
display charset = iso-8859-15
dos charset = 850
```

Da nun das AD mit Kerberos-Tickets arbeitet, muss der Kerberos-Client im Linux-System entsprechend eingerichtet werden. Analog zu der smb.conf müssen hier Realm und Rechnername des Controllers hinterlegt werden, damit die Authentifizierung stattfinden kann. Die Domain Controller finden sich hier als Key Server (kdc) wieder:

/etc/krb5.conf

```
[libdefaults]
     # default_realm = raum22.vhs-dt.de

...

[realms]
raum22.vhs-dt.de = {
     kdc = winserver1
     kdc = winserver2
     admin_server = winserver1
}
```

Wenn diese Einträge vorhanden sind, kann der Rechner in die Domäne integriert werden, was mit dem folgenden Befehl geschieht:

```
net join -S winserver1 -U Administrator
```

Der Benutzername muss nicht zwangsläufig Administrator lauten, der Account muss lediglich entsprechende Rechte zum Hinzufügen des Rechnerkontos in das ADS besitzen. Im AD-System sollte nach dieser Aktion bereits der Name des Rechners auftauchen.

6.4.5.2 Anpassung der Benutzerauthentifikation

Um nun die Authentifizierung eines Benutzers zu ermöglichen, der sich über den xdm oder kdm anmeldet, muss zunächst dem System bekannt gegeben werden, dass der **winbind**-Daemon ebenfalls eine Quelle für Benutzereinträge ist. Das geschieht in dieser Datei:

/etc/nsswitch.conf

```
...

passwd:         files winbind
group:          files winbind
shadow:         files winbind

...
```

Der Rechner sucht somit erst in den lokalen Dateien, und anschließend über den winbind im AD-System. Sofern der winbind-Daemon läuft, müsste ein „*getent passwd*“ bereits die AD-Benutzer zusätzlich anzeigen.

Da die Anmeldung über das „**P**luggable **A**uthentification **M**odule“-System erfolgt, muss diese ebenfalls angepasst werden. In den meisten Distributionen sind die Einträge in den folgenden Dateien notwendig:

/etc/pam.d/common-auth

```
auth    required      pam_mount.so
auth    sufficient    pam_winbind.so use_first_pass
auth    required      pam_unix.so nullok_secure use_first_pass
```

In dieser Datei wird sichergestellt, dass bei einer Anmeldung eines Benutzers **erst** das AD nach dem Benutzernamen gefragt wird. Schlägt die Überprüfung fehl, wird nicht abgebrochen ("*sufficient*"), sondern die Authentifikation in der Benutzerdatenbank im Linux-System mit demselben Passwort (*use_first_pass*) fortgesetzt. Allen Prüfungen vorangestellt ist pam_mount, um das - jetzt noch - bekannte Passwort zu speichern, damit im "*session*"-Bereich die Windows-Shares gemountet werden können.

/etc/pam.d/common-account

```
account    sufficient      pam_winbind.so
account    required        pam_unix.so
```

Analog zur Anmeldung wird bei der Accountierung (Rechteprüfung) erst das AD gefragt und nur bei einem Fehlschlag in der Linux-Umgebung weiter geprüft.

/etc/pam.d/common-session

```
session     optional     pam_mount.so
session     sufficient   pam_winbind.so
session     required     pam_unix.so
```

Bei der Zusammenstellung der Session wird "pam_mount" wieder verwendet, um die benötigten Shares zu mounten.

6.4.5.3 Einbindung der Shares

Wie bereits erwähnt wird für den Automatismus zum Mount der Windows-Shares das Modul „**pam-mount**“ verwendet. Diese wird quasi zwischen Eingabe des Passwortes durch den Benutzer und Authentifizierung durch den *winbind* geschaltet und mountet nach Vorgabe durch die Konfigurationsdatei die Wondows-Shares unter Angabe der Benutzerkennung. Welche Shares wie in das System eingehängt werden, legt die folgende Datei „**/etc/security/pam_mount.conf“** fest. Die Syntax dieser Datei orientiert sich am "mount"-Befehl und hat den Vorteil, dass sie im Gegensatz zur allgemeinen „/etc/fstab“ auch mit Variablen arbeiten kann, wie auch in Benutzerdefinierte Skripten verzweigen kann. Somit kann ein Äquivalent zu den Logon-Skripten unter Windows hergestellt werden, das die automatische Anbindung an (Windows-) Fileserver beim Anmeldevorgang übernimmt.

Im Folgenden Beispiel wird der Share **daten** des Servers **filesrv1** in das Unterverzeichnis **S** des anzumeldenden Benutzers so gemountet, wobei die Zugehörigkeit der Dateien zum Angemeldeten Benutzerkonto geordnet werden. Das gleiche passiert mit dem Share **forum**, der in das Unterverzeichnis **X** eingehängt wird.

/etc/security/pam_mount.conf

```
...

# volume <user> <type> <server> <volume> <mount point>
#        <mount options> <fs key cipher> <fs key path>

volume * smbfs filesrv1 daten /home/RAUM22/&/S
uid=&,gid=&,dmask=0775,workgroup=RAUM22,iocharset=iso8859-
15,codepage=cp850 - -

volume * smbfs filesrv1 forum /home/RAUM22/&/X
uid=&,gid=&,dmask=0775,workgroup=RAUM22,iocharset=iso8859-
15,codepage=cp850 - -

...
```

Da dieses Modul ebenfalls die Session überwacht, werden die Verbindungen aurtomatisch wie-

der getrennt, sobald die Session beendet wird (Abmeldung).

Allerdings ist diese Form des Mounts nur sinnvoll, wenn der betroffene Rechner als Einzelarbeitsplatz genutzt wird. Für Server ist diese Form ungeeignet, da nach dem Mount **jeder** auf diesem PC angemeldete Benutzer und der ersten Benutzerrechten auf diesen zugreifen kann.

6.5 Samba als PDC

Natürlich kann Samba auch ohne jegliche Windows-Server als „große Brüder“ verwendet werden. Die Konfiguration erfolgt dabei analog zu der Integration in eine NT-Domäne, nur dass die Referenz auf des NT-Server entfällt und alle Konten lokal abgebildet werden müssen.

6.5.1 Konfiguration in der smb.conf

Der erste Schritt ist die Umstellung des Parameters „*security*“ auf den Wert „user“, damit die lokalen Konten für die Autorisierung verwendet werden. Des weiteren muss der Server als Logon-Server („*domain logons*“) dem restlichen Netz entgegentreten, um den Clients zu ermöglichen, den für sie zuständigen DC ausfindig zu machen. Die notwendigen Veränderungen sähen folgendermaßen aus:

```
security = user
domain logons = yes
wins support = yes
preferred master = yes
local master = yes
domain master = yes
```

In diesem Zustand ist der Samba-Server schon fähig und in der Lage, als PDC zu arbeiten. Allerdings fehlen jetzt außer den Konten der Benutzer auch die Konten der PCs, die sich an dieser Domäne anmelden sollen. Aus der Sicht von Windows sind beides gleichberechtigte Konten, die sich nur durch ihren Zweck voneinander unterscheiden. Diese Eigenschaft wird beim Anlegen des Kontos dem Befehl „*smbpasswd*“ über die Option „-m“ mitgegeben. Wie bei einem Benutzerkonto muss für ein Maschinenkonto ein korrespondierender Eintrag in der /etc/passwd existieren, so dass an dieser Stelle leider zusätzliche Einträge entstehen. An den Namen des Rechners muss zusätzlich ein "$" gehängt werden, um das Konto als Maschinenkonto zu kennzeichnen. Soll also der PC mit dem Namen „PC01“ der Domäne hinzu gefügt werden, so geschieht das mit den Befehlen:

```
useradd -s /dev/false -d /dev/null pc01\$
smbpasswd -a -m pc01\$
```

Der dem "$" vorangestellte Backslash ("\") ist in diesem Fall notwendig, da die Shell anderenfalls das "$" als Variablen-Kennzeichnung missinterpretiert. Der Name selbst wird **klein** geschrieben, auch wenn Windows die Rechnernamen in Großbuchstaben verfasst.

Nun bietet Windows für die Eingliederung in eine Domäne auf dem jeweiligen Client an, diese Eintragung selbst vorzunehmen, sofern ein Domänen-Konto verfügbar ist, das über administrative Rechte verfügt (meistens „Administrator“). Auf diese Weise wird der zusätzliche Administrationsaufwand gespart und Tippfehler in der Namenswahl vermieden. Für diese Funktion wird der „*add user script*“ Eintrag wiederverwendet:

```
add user script = useradd -g ntuser -s /bin/false %u
```

Im Gegensatz zu der Mitgliedsserver-Funktion wird in diesem Kontext der angegebene Befehl dann ausgeführt, wenn vom Windows-Server aus ein Konto mit root-Rechten angegeben wird. Da der „root“ selbst für Samba-Logins nicht zulässig sein sollte, ist die Einrichtung eines spezi-

ellen Samba-Users für diese Zwecke sinnvoll.

6.5.2 Logon-Scripts

Ein Rechner, der sich an einem Server anmeldet, verwendet häufigeinige der Ressourcen, die dieser Server anbietet. Diese werden üblicherweise durch kleine Programme oder Befehlsskripten automatisch nach dem Anmeldevorgang ausgeführt. In der Windows-Welt wird dazu die Freigabe mit dem Namen „NETLOGON“ verwendet, in der sich die Skripten befinden. In der „*smb.conf*“ muss zunächst diese Freigabe eingerichtet werden:

```
[netlogon]
        comment = Logon Scripts
        path = /data/netlogon
        read only = yes
```

Das Skript selbst wird in der globals-Sektion bekannt gegeben:

```
logon script = logon.bat
```

Windows kann dabei jedem Benutzer-Account einen eigenen Skript-Namen zuordnen, so dass jeder Benutzer sein eigenes Skript benutzen könnte. Da Samba keine eigene Benutzerverwaltung verwendet, kann diese Information nicht direkt im Account gespeichert werden. Statt dessen kann aber die Fähigkeit „Wildcards“ zu interpretieren, genutzt werden. Da „%u“ für den jeweiligen Benutzernamen steht, kann der Wert der obigen Variable so ergänzt werden, dass der Name in den Dateinamen eingefügt wird:

```
logon script = logon\%u.bat
```

Auf diese Weise wird der Benutzer „meier“ das Skript „*logon\meier.bat*“ und der Benutzer „schmidt“ das Skript „*logon\schmidt.bat*“ benutzen. Dieser Weg mutet zwar etwas umständlich an, ist aber im Endeffekt nicht fehleranfälliger oder komplizierter als der Weg der Windows-Systeme. Weil diese Informationen direkt von de Client ausgewertet werden, müssen Pfadnamen in Windows-Syntax angegeben werden („\“ statt „/“).

6.5.3 User-Profile

Wenn schon Logon-Skripten möglich sind, ist es ein Leichtes, zu diesem Zweck auch noch die Sicherheitsrichtlinien abzulegen. Allerdings wird hierfür das Programm „ntprofile.exe“ benötigt, das lediglich von einem Windows-NT-Server zu beziehen ist. Mit diesem Tool wird eine Datei „ntconfig.pol“ erzeugt, die im Logon-Verzeichnis abgelegt wird. Mit diesem Tool können bestimmte Berechtigungen ein- oder ausgeschaltet werden, die in der beschriebenen Datei festgehalten werden. Meldet sich ein NT- oder Windows 2000- Rechner an dem Samba-Server an, wird diese Datei gelesen und automatisch in die aktuelle Session übertragen.

Leider gibt es noch kein Tool, das in die Konfiguration dieser Datei unter Linux ermöglicht oder gar in den SWAT einbindet.

6.6 Komfortable Konfiguration

Wie so ziemlich jedes Software-System in der UNIX-Welt, kann auch Samba allein mit einem Editor vollständig konfiguriert werden. Nun ist aber die manuelle Veränderung einer Konfigurationsdatei nicht nur unkomfortabel, sie ist auch Fehlerträchtig. Wird ein Parameter falsch geschrieben, oder die Syntax nicht korrekt eingehalten,. können die Resultate unter Umständen erst im laufenden System bekannt werden, sofern der Server überhaupt läuft.

Um zumindest die Korrekte Schreibweise garantieren zu können, wurde SWAT (Samba Web Administration Tool) entwickelt, das auf einfache Art und Weise eine komfortable Konfigurationsoberfläche anbietet. Aus der Erfahrung heraus werden Server nicht als Arbeitsplatzrechner

benutzt, sondern fristen ihr Dasein mehr oder weniger still in einer Ecke eines separaten Serverraumes. Daher wäre es wenig sinnvoll gewesen ein grafisches Tool zu bauen, das die Anwesenheit des Administrators vor Ort am Serverrechner erfordert.

Statt dessen wurde ein minimaler Web-Server gebaut, der mit einem beliebigen Browser bedient werden kann. Auf diese weise ist nicht nur der Server von der Ferne aus konfigurierbar, das Konfigurationstool ist sogar weitestgehend Plattform-unabhängig. Normalerweise ist jedoch der SWAT aus Sicherheitsgründen deaktiviert. Um dieses zu ändern, muss die Datei */etc/inetd.conf* geändert werden, indem die folgende Zeile hinzugefügt oder von ihrem Kommentarzeichen ('#') befreit wird:

```
swat    stream  tcp     nowait.400      root     /usr/sbin/swat
swat
```

Nachdem der *inetd* neu gestartet wurde, kann von beliebiger (erreichbarer) Stelle im Netz aus die Konfiguration über die URL **http://*<rechnername>*:901** erreicht werden. Die darauf folgende Autorisierung verlangt die Kennung "root" mit dem gültigen Passwort.

Ist die Anmeldung erfolgreich, erscheint die Übersichtsseite des SWAT, die mit einer schieren Unmenge von Dokumentation aufwartet. Die obere Leiste bietet die eigentlichen Konfigurationsbereiche an, die sich in „*globals*“, „*shares*“ und „*printers*“ unterteilen. Diese Unterteilung spiegelt die Bereiche der „*smb.conf*“ wieder, wobei die einzelnen Optionen und Variablen in Form von Textfeldern oder Auswahllisten angeboten werden.

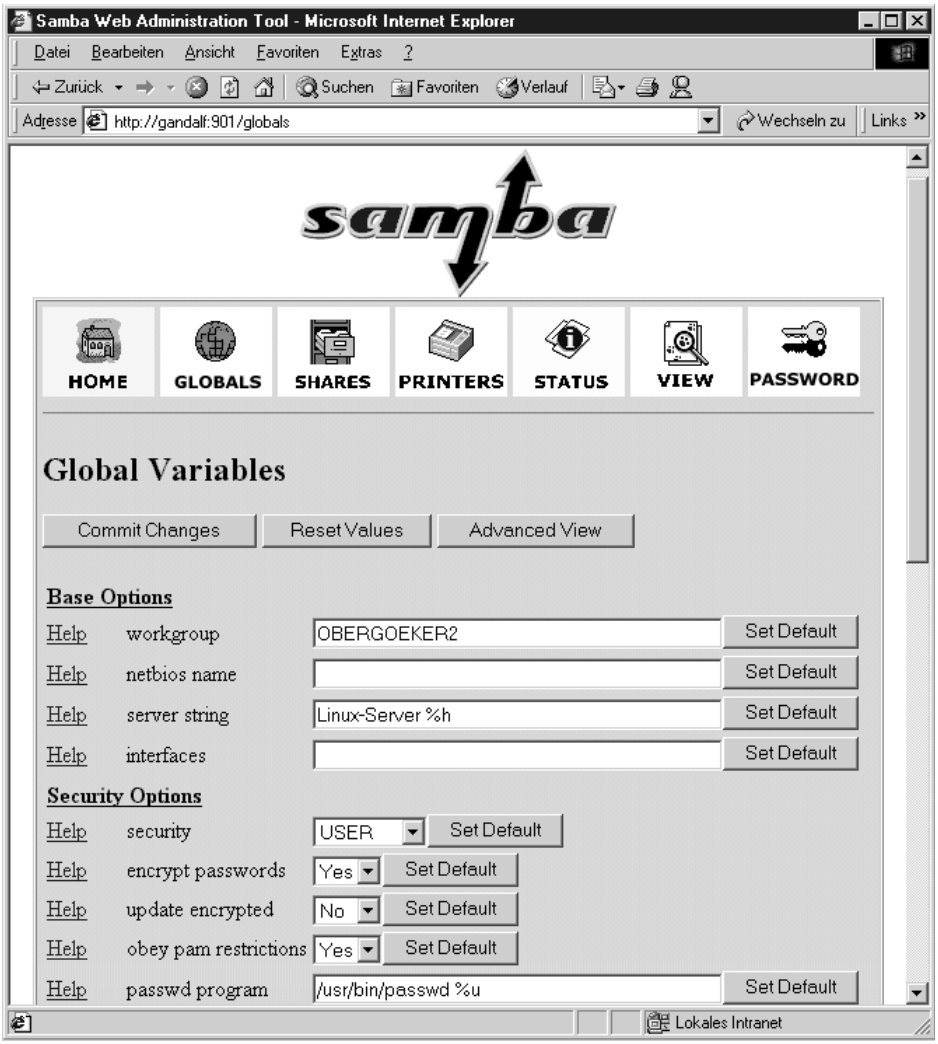


Abbildung 7: Konfiguration mit SWAT

Werden Änderungen durchgeführt, müssen diese zwingend mit dem Knopf „*Commit Changes*“ gesichert werden, da sie sonst nicht Abgelegt werden. Standardmäßig werden nur die Optionen angezeigt, die obligatorisch sind oder nicht dem Defaultwert entsprechen. Sollen **alle** Optionen angezeigt werden, kann der Knopf „*Advanced View*“ betätigt werden.

Diese Form der Konfiguration hat den Vorteil, dass keine Syntaxfehler entstehen können, aber andererseits können Kommentare nicht in der Datei berücksichtigt werden. Unschlagbar ist allerdings, dass zu **jeder** Option ein Link auf die korrespondierende Stelle in der Samba-Dokumentation existiert, so dass jederzeit auf die zur Samba-Version passende Hilfe zugegriffen werden kann.

7 Mars-NWE

Der Mars-NWE (Martin Stoevers Netware-Emulator) ist ein Daemon, der optional gestartet werden kann. Die Funktionsweise ist der des Samba-Servers nicht unähnlich, jedoch setzt seine Netzwerk-Fähigkeit auf einem anderen Protokoll auf, dem SPX/IPX. Das SPX/IPX ist eine optionale Komponente, die zwar eingebunden werden kann, doch immer einen Kompromiss zwischen Funktionalität und Kompatibilität bildet. Deshalb sollte der NWE auch nur dann eingesetzt werden, wenn eine vorhandene Novell Netware-Landschaft um einen Linux-Rechner erweitert werden soll. Allerdings ist schon oft beobachtet worden, dass die Datenübertragung mittels SPX/IPX um einiges schneller ist als über SMB (Samba) oder NFS. Das Netware-Konzept der Datei- und Userverwaltung basiert auf einer völlig anderen Philosophie und ist deshalb nur mit Umwegen in das Linux-Konzept einbindbar. Wird ein völlig neue Netzstruktur entworfen, sollte dem Samba-Server der Vorzug gegeben werden, zumal die Client-Software für (reine) DOS-PCs von Microsoft frei verfügbar ist.

Der MARS-NWE unterstützt folgende Funktionen:

Fileserver-Dienste: Die Hauptaufgabe des NWE-Server ist die Bereitstellung von Festplattenkapazität, wobei diese hauptsächlich auf DOS-Rechner ausgerichtet ist. Es werden daher standardmäßig keine langen Dateinamen unterstützt, sodass alle Bezeichnungen dem "8.3"-Standard entsprechen müssen. Auch muss die Groß-/Kleinschreibung nur auf eine Art umgesetzt werden, indem entweder alles in Kleinbuchstaben oder in Großbuchstaben erwartet und erkannt werden kann. Allerdings können beliebige Dateisysteme exportiert werden, um z.B. ein über NFS gemountetes UNIX-Filesystem auch DOS-PCs anbieten zu können.

Printserver-Dienste: Die Druckerfreigabe kann auch auf die Linux-lpr-Dienste abgebildet werden, Allerdings besteht noch eine Schwierigkeit mit Netware-DOS-Clients einer Version (4.0. Dieses Umfeld birgt Probleme mit den Printer-Queues, die dazu führen, dass Druckaufträge nicht direkt ausgeführt werden, sondern erst dann, wenn die NWE-Verbindung vom Client geschlossen wird.

Routing: Eine der besonderen Fähigkeiten ist das automatische Routing, wenn der Linux-Rechner ohnehin als Gateway/Router zwischen zwei Netzwerken eingesetzt wird. In diesem Fall können beide IPX/SPX-Netze zu einem zusammengeschlossen werden, was besonders bei einer Modemverbindung interessant wird. Allerdings muss zu diesem Zweck zwingenderweise PPP eingesetzt werden, weil das SLIP keine anderen Protokolle als IP unterstützt.

PIPE-Volumes: Um das Problem der fehlenden RPC-Fähigkeiten des IPX/SPX zu umgehen, stellt der NWE als besondere Fähigkeit Volumes zur Verfügung, die nach Außen wie normale Dateien erscheinen, die aber intern nur die Input-Queue für beliebig definierbare Prozesse sind. Allerdings konnte die Funktionalität dieser PIPE-Verzeichnisse bislang nicht unter Netware 4.0 aktiviert werden

Um den Mars-NWE nutzen zu können, müssen zunächst einmal Im Kernel in der Sektion "Networking options" die abgebildeten Einträge folgendermaßen konfiguriert werden:

```
<M> The IPX protocol
[ ] IPX: Full internal IPX network
```

Anschließend muss der Kernel natürlich neu kompiliert, installiert und neu gebootet werden, bevor der Mars-NWE installiert werden kann.

Sofern der Mars-NWE nicht bereits mit der jeweiligen Linux-Distribution mitgeliefert wurde, ist der "Mars-NWE" unter

http://www.compu-art.de/marsnwe1.htm

und auf allen guten Linux-Mirrors zu finden, wie z.B.

ftp://ftp.gwdg.de/pub/linux/misc/mars_nwe

Üblicherweise wird er als *.tgz-Paket ausgeliefert, weswegen er zunächst mit dem Befehl

```
tar -xvzf mars_nwe*tgz
```

entpackt werden muss. In diesem Verzeichnis müssen dann mit root-Rechten folgende Befehle in gleicher Reihenfolge ausgeführt werden:

- "make" zur Ermittlung der aktuellen Umgebung
- "config.h" mit einem Editor (vi) überprüfen und ggf. anpassen
- "make" ein weiteres Mal aufrufen, wobei dieser Aufruf zu der Kompilierung des NWE führt.
- "nw.ini" mit einem Editor auf den eigenen Gebrauch abstimmen
- "make install" aufrufen, damit die Binaries und die Konfigurationsdateien an die richtige Stelle im System kopiert werden.
- "nwserv" starten

Es ist dringenst davon abzuraten, den nwserv zu starten, bevor nicht die Datei "/etc/nwserv.conf" angepasst worden ist, ansonsten können unbeabsichtigte Sicherheitslöcher entstehen.

7.1 Konfiguration

Die Konfiguration des Mars-NWE erfolgt über die Datei "/etc/nwserv.conf" die für Mars-NWE eine ähnliche Bedeutung wie die "/etc/smb.conf" für Samba. Auch hier werden freigegebene Verzeichnisse, Drucker und allgemeine Sicherheitsbestimmungen festgelegt. Im Unterschied zu Samba werden jedoch zusätzlich noch die definierten Benutzer, deren Mapping zu den Linux-Usern und - möglicherweise - sogar deren Passwörter im Klartext gespeichert! Aus diesem Grund darf diese Datei ausschließlich für den root lesbar sein und muss vor Benutzung des Systems mit "chmod 600" geschützt werden. Trotz dieser zentralen Datei können nicht alle Optionen hierüber eingestellt werden, ggf. müssen bestimmte Verhaltensweisen in der "config.h" (s.o.) eingestellt und das Paket neu übersetzt werden.

Die Datei nwserv.conf ist in nummerierte Abschnitte (sections) unterteilt, die jeweils eine eigene Bedeutung besitzen Diese Abschnitte sind nicht durch Überschriften unterteilt, sondern jede Zeile wird durch eine vorangestellte Abschnittsnummer einem solchen zugeordnet; es dürfen keine Zeilen ohne Abschnittsnummer existieren Ein Kommentar wird durch ein vorangestelltes "#" gekennzeichnet, alle Zahlen werden - bis auf wenige Ausnahmen - als Dezimalwerte angesehen. Hexadezimalzahlen (bzw. Sedezimalzahlen) müssen durch ein vorangestelltes "0x" gekennzeichnet werden. In der Standardversion wird diese Datei mit sehr viel Kommentaren ausgeliefert, die detailliert die benutzten Parameter beschreiben. Der Übersichtlichkeit halber sollen die wichtigsten Parameter hier noch einmal (natürlich in deutscher Sprache) beschrieben werden

Section 1

In der ersten Sektion werden die wichtigsten Einstellungen vorgenommen: Die freigegeben "Laufwerke", wie sie von DOS aus mit dem "map"-Befehl benutzbar sind. jedes Laufwerk wird auf ein beliebiges Verzeichnis im Linux-System abgebildet, wobei der Name dieses nicht irgendwelchen Konventionen entsprechen muss. Innerhalb dieses Verzeichnisses müssen jedoch alle Dateien und Verzeichnisse dem DOS-Standard entsprechen, sonst bleiben sie für die Cli-

ents einfach "unsichtbar". Da das Verzeichnis "SYS" eine besondere Rolle in der Netware-Welt spielt, erstellt es der Daemon "nwserv" automatisch sowie die Unterverzeichnisse "PUBLIC", "SYSTEM" und "MAIL", die von den Clients für den Login-Vorgang benötigt werden.

Alle weitere Verzeichnisse sind optional, prinzipiell kann auch ausschließlich mit dem "PUBLIC"-Verzeichnis gearbeitet werden. Die Optionen legen bei den freigegeben Verzeichnissen die Verhaltensweise fest:

i Groß-/Kleinschreibung wird ignoriert. Alle Dateien werden für die Clients in einer einheitlichen Form Dargestellt. Da dieser Parameter Performance-Einbußen mit sich bringt, sollte er nur dann eingesetzt werden, wenn er wirklich gebraucht wird.

k Alle Dateinamen werden in Kleinbuchstaben erwartet. wird dieses Flag nicht gesetzt, müssen alle Dateien in Großbuchstaben gesetzt werden, um sie für die Clients benutzbar zu halten

m Entfernbares Laufwerk. Das Verzeichnis ist z.B. ein CDROM-Mount, der während der Laufzeit gewechselt werden kann.

n Keine festen Inodes. Dieses Flag muss gesetzt werden, wenn das dahinterliegende Filesystem keine Inodes besitzt, Wie z.B. DOS- (FAT-) Partitionen, CDROMs etc.

o (kleines "o") Das Verzeichnis enthält nur 1 Filesystem

p "PIPE"-Filesystem. Alle hier erscheinenden Dateien werden als PIPE-Kommados interpretiert

r Read-Only-Verzeichnis

t Verzeichnis benutzt "trustees", wie sie unter "/var/nwserv/trustees" abgelegt werden. . Betrifft aber nur Dateisysteme ohne feste Inodes (s. Option "n")

O (großes "O") Das Verzeichnis wird als OS/2-Namespace abgebildet, in dem auch lange Rechner- und Freigabenamen unterstützt werden. Diese Eigenschaft betrifft allerdings nicht nur OS/2- sondern auch Windows 9x, und Windows NT-Clients.

N Verzeichnis benutzt NFS-Namespace (selten benutzt).

Nachfolgend werden noch die "umask"-Einträge für neu erstellte Verzeichnisse und Dateien angehängt, die aber optional sein können. Zu beachten ist allerdings, dass nicht die umask-Zahlenkette aus dem UNIX-Standard angegeben wird, sondern die Oktalzahlenfolge, die die erstellten Dateien tatsächlich besitzen. Im nachfolgenden Beispiel wird außer dem "SYS"-Laufwerk auch ein "DATA"-Laufwerk definiert, das für alle angemeldeten Benutzer les- und beschreibbar ist und in dem auch nur Dateien angelegt werden, die für alle Benutzer vollen Zugriff bieten (umask 666)

```
# Section 1: volumes (required)
# Syntax:
# 1  VOLUMENAME  DIRECTORY  [OPTIONS]  [UMASKDIR UMASKFILE]
1   SYS    /usr/local/nwe/SYS/     kt   711 600
1   DATA   /data                   kt   777 666
```

Section 2

In der 2. Sektion wird der Servername festgelegt. Wird er nicht angegeben, nimmt nwserv automatisch den IP-Hostnamen des Rechners.

```
# Section 2: servername (optional)
# Syntax:
# 2  SERVERNAME
```

```
2    LINUX
```

Section 3

An dieser Stelle wird die interne Netzwerknummer angegeben. Der Eintrag darf nicht fehlen, weil IPX ähnlich wie TCP/IP eine Netzwerknummer der Knotennummer voranstellt. Die Knotennummer ist prinzipiell die 6-Byte MAC der Ethernet-Karte, die (interne) 4-Byte Netzwerknummer ist frei wählbar. Ein internes Netz besteht immer aus einem Server und den angeschlossenen Clients, daher muss diese Netzadresse pro Server eindeutig sein. Auch wenn mit dem Parameter "auto" eine Nummer aus der aktuellen IP-Adresse generiert wird, sollte sich eine Abstimmung mit anderen Netware-Servern erfolgen, sofern welche vorhanden sind.

```
# Section 3: Number of the internal network (required)
# Syntax:
#       3       INTERNAL_NET    [NODE]
3       auto
```

Section 4

Die IPX-Devices müssen zwingenderweise konfiguriert werden, weil der Protokoll-Frame auf dieser Ebene festgelegt wird. An dieser Stelle wird auch der eingebaute IPX-Router Konfiguriert, weil mehrere Devices, also auch ppp-Interfaces festgelegt werden können. Als Frames werden folgende Protokolle bereitgestellt:

- **ethernet_ii**:Wird vor allem bei mehreren Netzprotokollen (IP + IPX) verwendet
- **802.2**: (IEEE 802.2) Default-Protokoll von Novell ab der Version 3.12
- **802.3**: (IEEE 802.3) Vorgänger von 802.2, wird immer noch von einigen BOOT-ROMs benutzt
- **snap**: In älteren Netzwerken mit mehreren Protokollen (IPX + IP) in Verbindung mit Token Ring benutzt
- **token**: Token Ring
- **auto**: Einfachste Entscheidung: nwserv ermittelt selbst das Protokoll

Die folgende Einstellung ist so ziemlich für alle Netzwerke gültig.

```
# Section 4: IPX-devices (strongly recommended)
# Syntax:
# 4  NET_NUMBER   DEVICE  FRAME   [TICKS]
4   0x22    eth0    ethernet_ii  1
```

Section 5

Diese Sektion enthält eine Option, die für das IPX-Routing relevant ist. eine "0x1" lässt alle Routen permanent erscheinen, sodass sie auch dann zur Verfügung stehen, wenn der nwserv zwischenzeitlich gestoppt wird. "0x2" lässt den Kernel entscheiden, ob die eingetragenen Routen noch benötigt werden, was jedoch teilweise Probleme mit Windows 9x erzeugt. "0x4" schließlich löscht alle Routen beim Beenden von nwserv.

Der Default wird mit "0x0" eingestellt, was z.Zt. noch derselbe wie "0x1" ist

```
# Section 5: special device flags
5       0x0
```

Section 6

Da sich der Linux-Server auch mit einer Versionsnummer im IPX-Netz melden muss, kann die-

se an dieser Stelle vorgegeben werden. Vorgesehen sind dafür 3 verschiedenen Werte:

- **0**: Version 2.15
- **1**: Version 3.11
- **2**: Version 3.12

Die bevorzugte Version ist momentan die 3.11. Achtung: Eine Veränderung der Versionsnummer beeinflusst nicht das Verhalten des Servers, lediglich die Clients werden unterschiedliche Routinen für den Serverzugriff benutzen.

```
# Section 6: version-"spoofing"
# Syntax:
# 6       SERVER_VERSION  [FLAGS]
6       1   0x0
```

Section 7

Die Passwörter werden zwar von nwserv ohnehin in verschlüsselter Form abgelegt, nicht jedoch in dieser Form vom Client übertragen. Wird dieses Flag auf "0x0" gesetzt, werden die Client-Komponenten zusätzlich veranlasst, die Passwörter vor dem Versenden zu verschlüsseln. Außerdem wird verhindert, dass Passwörter "leer" sein können. Das Gegenstück wäre der Wert "0x8", der sowohl unverschlüsselte als auch leere Passwörter erlaubt. Es sollte also nur dann ein anderer Wert als "0x0" eingesetzt werden, wenn es unbedingt erforderlich ist.

```
# Section 7: password handling of DOS-clients (required)
# Syntax:
# 7       Value
7       0
```

Section 8

In dieser Sektion wird das Verhalten des Servers in sicherheitsrelevanten Fällen festgelegt. Da gerade DOS-Clients keine Rechte kennen, müssen diese auf die Ebene der Dateiattribute übertragen werden. Die einzuschaltenden Optionen gliedern sich folgendermaßen auf:

- **0x1**: Es werden auch Dateien aufgelistet, die sich nicht im \SYS\LOGIN-Verzeichnis befinden, sofern der betreffende User angemeldet ist
- **0x2**: Dieser Parameter veranlasst nwserv dazu, eine schreibgeschützte Datei zunächst ohne Fehlermeldung mit r/w-Zugriff zu öffnen und erst beim Schreibversuch einen Fehler auszugeben.
- **0x4**: Ist dieser Wert aktiv, erlaubt nwserv das Umbenennen von Verzeichnissen, was normalerweise zu einem Fehler führt.
- **0x8**: Normalerweise darf sich ein Supervisor nur an bestimmten Stationen und zu bestimmten Zeiten anmelden. Ist dieser Wert eingestellt, wird diese Beschränkung aufgehoben.
- **0x10**: Über diesen Wert kann eine Datei auch dann gelöscht werden, wenn sie von einem anderen Prozess geöffnet ist.
- **0x40**: Manche DOS-Clients kommen mit Plattengrößen > 2GByte nicht zurecht. Mit Hilfe dieses Wertes wird der (angezeigte) freie Platz nie mehr als 2 GByte betragen.

Dieser Parameter sollte auf "0x0" bleiben und nur im Bedarfsfall umgestellt werden.

```
# Section 8: special login/logout/security and other flags.
8       0x0
```

Section 10 und 11

Durch den Umstand, dass sich (DOS-) Rechner schon mit dem Server verbinden (attach) , bevor sie angemeldet sind, erfordert einen Benutzer mit minimalen Rechten. Unter Linux ist dieser User "nobody" mit der Gruppe "nogroup" Beide besitzen den numerischen Wert 65534, den höchstmöglichen Wert für eine User-ID. Mit dieser Berechtigung wird der Inhalt des Verzeichnisses "\SYS\LOGIN" angezeigt, wenn sie entsprechend in der Sektion 10 und 11 hinterlegt wird. Fehlt sie, kann von einem DOS-Rechner kein Login stattfinden.

```
# Section 10: UID and GID with minimal rights
# Syntax:
# 10      GID
# 11      UID
10      65534
11      65534
```

Section 12

Der Mars-NWE besitzt ein eigenes Benutzerkonzept und hält auch die Informationen über diese Benutzer in einer eigenen Bindery-Datei, arbeitet also ähnlich wie Samba in Verbindung mit der "smbpasswd". Nur können die Bindery-Dateien weder manuell editiert werden, noch können sie mittels eines Linux-Werkzeuges wie "passwd" vorgegeben werden. Daher muss das Passwort einmal im Klartext in der "/etc/nwserv.conf" abgelegt werden, bevor der Server das erste Mal startet. Danach kann und muss es ohne Probleme wieder entfernt und mittels SETPASS.EXE vom Client aus geändert werden. An dieser Stelle wird ein Sicherheitsloch in der Mars-NWE-Konfiguration deutlich, denn wenn diese Datei nicht gegen den Zugriff anderer gesichert wird, kann das Passwort im Klartext eingesehen werden. Der 2. Parameter gibt den Linux-User an, auf den der Netware-Account übertragen wird. Im einfachsten Fall ist das der "root", doch sollte abgewogen werden, ob nicht für diesen Zweck ein explizit erstellter User besser geeignet ist.

```
# Section 12: supervisor-login (required)
# Syntax:
# 12    NW_LOGIN     LINUX_LOGIN     [PASSWORD]
12  SUPERVISOR   root    mars
```

Section 13

Analog zur Sektion 12 werden her die Accounts der anderen User festgelegt. Auch hier werden die Linux-User als 2. und das vorgegebene Passwort als 3. Parameter (temporär!!) mitgegeben. Jeder Netware-Benutzer muss einen gültigen Account auf der Linux-Seite besitzen, oder er wird automatisch auf den User gemapt, der in den Sektionen 10 und 11 definiert wurde. Natürlich könnte an dieser Stelle statt in der Sektion 12 ein User "Supervisor" festgelegt werden, doch hätte er nicht die entsprechenden Rechte zur Administration des Systems.

```
# Section 13: user-logins (optional)
# Syntax:
# 13  NW_LOGIN  LINUX_LOGIN [PASSWORD]  [FLAGS]
13  SCHULZ  schulz
13  MEIER_F  meier
```

Section 15

Um den Aufwand der manuellen Definition der Benutzer zu umgehen, können die im Linux-System ohnehin vorhandenen User auf das Mars-NWE-System übertragen werden. Es werden

allerdings nur die User übertragen, die sich auch unter Linux direkt anmelden können, also kein "*" im passwd-Eintrag der "/etc/shadow" vorweisen.

Da jedoch die vorhandenen Passwörter nicht übernommen werden können, muss ein "default"-Passwort definiert werden, das auf alle User übertragen wird, die neu hinzukommen. Da dieses Passwort - besonders in empfindlichen Bereichen - niemandem mitgeteilt werden sollte, muss der Administrator zunächst sich selbst unter diesem Account anmelden und das vorhandene Passwort ändern.

Folgende Optionen sind gültig:

- Keine automatische Übertragung
- Übertragung wird durchgeführt, und alle unbekannten Linux-User mit dem Default-Passwort in das Mars-NWE-System übertragen
- Bei allen Usern wird das Passwort auf das Default-Passwort gesetzt.

Im Zweifel sollte dieser Wert auf "0" bleiben, und nur bei wirklich vielen Usern (>30) auf "1" gesetzt werden.

```
# Section 15: automatic mapping of logins
# Syntax:
# 15      FLAG    DEFAULT_PASSWORD
15   0     top-secret
```

Section 21

Nach der Sektion 1 ist diese die zweitinteressanteste, weil hier die benutzbaren Drucker angegeben werden. Die Definition ist dabei ähnlich der in Sektion 1. Der nach Außen sichtbare Name ist der 1- Parameter (Nach der Sektionsnummer), der 2. Parameter ist das Spool-Verzeichnis, das normalerweise durch den Eintrag "-" automatisch im "SYS"-Laufwerk erzeugt wird.

Der wichtigste Teil ist jedoch der "Print Command", der die eigentliche Aktion darstellt. Soll z.B. der Standarddrucker benutzt werden, reicht ein schlichtes "lpr", es können aber alle Variationen benutzt werden, die der "lpr" unter Linux hergibt. Auf diesem Weg können daher auch entfernte Drucker angesprochen werden, sofern sie den lpd unterstützen. Der DOS- Client ""sieht" dabei nur den Drucker als normalen Netware-Drucker.

```
# Section 21: print queues (optional)
# Syntax:
# 21      QUEUE_NAME      [QUEUE_DIR]       [PRINT_COMMAND]
21     HP500      -    lpr -Plp
```

Section 22

Soll im Gegensatz zu der vorigen Section ein echter Netware-Printserver genutzt werden, wie z.B. ein mit "PSERVER" freigegebener Drucker einer DOS-Arbeitsstation, so kann er in dieser Section definiert und somit allen Clients zur Verfügung gestellt werden. Der Aufbau ist denkbar einfach: Es muss lediglich erst der Stationsname und anschließend der Name der Printer-Queue angegeben werden, um die Daten dorthin weiterleiten zu können.

```
# Section 22: print server entries (optional)
# Syntax:
#       22 PSERVER_NAME  QUEUE_NAME  [FLAGS]
22    PS1    OCTOPUSS
```

Section 60 - 63

Diese Section kann als der "Traum" jeden Netware-Administrators bezeichnet werden, weil die

maximal zulässigen Verbindungen und somit die maximale User-Anzahl durch eine Veränderung in Section 60 beliebig hochgesetzt werden können. Die maximale Anzahl der Volumes in Section 62 ist allerdings selten zu ändern, weil 10 durchaus eine recht unübersichtliche Anzahl der Serverlaufwerke darstellt. Statt eine Unmenge Laufwerke zu definieren, ist es effektiver, wenige sinnvoll zu strukturieren. um auch die Wartbarkeit gewährleisten zu können.

Der Punkt 63, der die maximale Anzahl der "Base-Directorys angibt, ist mit 50 durchaus ausreichend, da nur die max. Anzahl der Hauptverzeichnis-Einträge gemeint ist. Die Anzahl hier sehr hoch zu setzen verführt nur zu einer unübersichtlichen Benutzung des Verzeichnisses, wenn extrem viele einzelne Dateien abgelegt werden.

```
# Changing defaults from config.h
# more information in config.h
60  10          # MAX_CONNECTIONS
61  10          # MAX_NW_VOLS
63  50          # MAX_DIR_BASE_ENTRIES
```

7.2 Beispielkonfiguration

Zum Schluss sei an dieser Stelle noch die vollständige "/etc/nwserv.conf" abgebildet, wie sie nach en obigen Beispielen aussehen müsste. Die komplette Datei enthält allerdings eine Reihe weiterer, im Text nicht weiter beschriebene Sektionen, die aber bewusst ignoriert wurden, da die meisten lediglich für Debug-Informationen oder zum Fine-Tuning verwendet werden. bei ihnen ist es ratsam, die vorgegebenen Werte zu übernehmen. Ausnahme sind die Sektionen 200-202, weil die Position der Logfiles nach Distribution oder Geschmack variieren kann.

```
# Section 1: volumes (required)
# Syntax:
# 1  VOLUMENAME  DIRECTORY  [OPTIONS]  [UMASKDIR UMASKFILE]
1   SYS    /usr/local/nwe/SYS/     kt   711 600
1   DATA   /data                   kt   777 666

# 2  SERVERNAME
2   LINUX

# Section 3: Number of the internal network (required)
# Syntax:
#       3       INTERNAL_NET    [NODE]
3       auto

# Section 4: IPX-devices (strongly recommended)
# Syntax:
# 4  NET_NUMBER   DEVICE  FRAME   [TICKS]
4   0x22    eth0    ethernet_ii  1

# Section 5: special device flags
5       0x0

# Section 6: version-"spoofing"
# Syntax:
# 6       SERVER_VERSION  [FLAGS]
6       1   0x0

# Section 7: password handling of DOS-clients (required)
# Syntax:
# 7       Value
7       0

# Section 8: special login/logout/security and other flags.
```

```
8       0x0

#
9 0751  0640

# Section 10: UID and GID with minimal rights
# Syntax:
# 10      GID
# 11      UID
10      65534
11      65534

# Section 12: supervisor-login (required)
# Syntax:
# 12    NW_LOGIN     LINUX_LOGIN     [PASSWORD]
12  SUPERVISOR   root    mars

# Section 13: user-logins (optional)
# Syntax:
# 13  NW_LOGIN  LINUX_LOGIN [PASSWORD]  [FLAGS]
13  SCHULZ  schulz
13  MEIER_F  meier

# Section 15: automatic mapping of logins
# Syntax:
# 15      FLAG    DEFAULT_PASSWORD
15   0     top-secret
16      1
17       0x0
18       0x0

# Section 21: print queues (optional)
# Syntax:
# 21      QUEUE_NAME      [QUEUE_DIR]       [PRINT_COMMAND]
21     HP500      -    lpr -Plp

# Section 22: print server entries (optional)
# Syntax:
#       22 PSERVER_NAME  QUEUE_NAME  [FLAGS]
22    PS1    OCTOPUSS

#  40 = path for vol/dev/inode->path cache
40  /var/spool/nwserv/.volcache
#  41 = path for share/lock files
41  /var/spool/nwserv/.locks
#  42 = path for spool dir
42  /var/spool/nwserv

#  45 = path for bindery file's
45  /var/nwserv/db
#  46 = path for attribute handling
46  /var/nwserv/attrib
#  47 = path for trustee handling
47  /var/nwserv/trustees

# Changing defaults from config.h
# more information in config.h
60  10      # MAX_CONNECTIONS
61  10      # MAX_NW_VOLS
63  50      # MAX_DIR_BASE_ENTRIES
```

```
100     0   # debug IPX KERNEL (0 | 1)
101     1   # debug NWSERV
102     0   # debug NCPSERV
103     0   # debug NWCONN
104     0   # debug NWCLIENT, should *always* be '0' !
105     0   # debug NWBIND
106     1   # debug NWROUTED

# Sections 200-202: logging of "nwserv"
#
200     1   # 0 = no logfile
            # 1 = daemonize nwserv/nwrouted and use logfile
201     /var/log/nw.log # filename of logfile
202     0x1 # & 0x1=creat new logfile

# Sections 210,211: timing

210     5   # 1 .. 600  shutdown-delay in sec.
211     60  # 10 .. 600 broadcasts every x seconds

# Sections 300-302: loging of routing-information

300     1
301     /var/log/nw.routes
302     0x1

# Section 310: watchdogs
310     7

# Section 400:
# station file for special handling of stations.

400  /etc/nwserv.stations
# Section 401: nearest server
#
# for special handling of the 'get nearest server request'.
401     0

# Section 402: station connect restrictions
402     0
```

7.3 Client-Konfiguration

Die Konfiguration der Clients geschieht analog der Beschreibung von Netware. Windows95 und Windows-NT-Systeme sind hier im Vorteil, da sie schon alle nötige Software enthalten, um die freigegeben Ressourcen des Servers zu nutzen. An dieser Stelle sei noch einmal darauf hingewiesen, dass für eine reine Windows-Landschaft der Einsatz von Samba in Verbindung mit den MS-Lanmanager- (= SMB-) Komponenten lohnender und vorteilhafter ist.

Das Haupteinsatzgebiet von Mars-NWE sind jedoch DOS-und ggf. OS/2-Landschaften, die bereits mit Netware-Fileservern verbunden sind. Denn obwohl bereits Windows 95 und Windows NT eigene im Betriebssystem enthaltene Client-Software mitliefern, ist es doch wichtig zu erwähnen, dass die DOS-Software keinesfalls frei ist, und diese nur von Novell selbst bezogen werden darf.

Sobald ein DOS-Client die benötigten Treiber geladen hat, ist er an dem Novell-Server "attached", der in der net.cfg des Novell-Verzeichnisses als "PREFERRED SERVER" eingetragen ist. Wenn dieser der Linux-Rechner sein soll, ist dieser Eintrag natürlich entsprechend zu ergänzen. In diesem Zustand ist bereits ein Laufwerk dem Verzeichnis "LOGIN" auf dem Netware-

Volume "SYS" verbunden. Abhängig von dem Eintrag "LASTDRIVE" in der "CONFIG.SYS"-Datei ist das meistens das Laufwerk "F:", in dem sich die notwendigen DOS-Programme zum Anmelden an den entsprechenden Server befinden. Aus diesem Grund muss dafür gesorgt werden, dass die Datei "Login.exe", "Slist.exe" und ggf. noch "Map.exe" in dem Verzeichnis vorhanden sind. Im einfachsten Fall können sie von einem anderen Server kopiert werden, wobei aber die Lizenzbestimmungen nicht außer Acht gelassen werden dürfen. Außerdem müssen dieselben Dateien zusätzlich im Verzeichnis "PUBLIC" untergebracht werden, wobei mindestens noch "Capture.exe", "Printcon.exe" und "Pconsole.exe" hinzugefügt werden sollten. Sofern allerdings bereits andere Novell-Server im Netzwerk existieren, ist es einfacher, einen dieser Server als "preferred Server" einzutragen. Ab diesem Punkt steht der Linux-Rechner als Netware-Server zur Verfügung. Sämtliche Konfigurationen können dann unter dem Login "Supervisor" durchgeführt werden, jedoch müssen Veränderungen am Server wie z.B. neue Verzeichnisfreigaben, Benutzerverwaltung, Anschluss neuer Drucker etc. nach wie vor unter einem Linux-Account als "root" durchgeführt werden.

Zurzeit wird außerdem eine Alternative zu den vorgenannten Netware-DOS-Programmen erarbeitet, die von demselben Autor stammt, und auch unter der GPL steht: Die Mars-Dosutils. Sie bestehen prinzipiell aus einem einzigen Programm, das - über Parameter gesteuert - alle Aufgaben übernehmen kann, die mit den Novell-eigenen Mitteln gelöst werden müssten. Lt. Autor sind diese jedoch noch "very incomplete".

7.4 Probleme

Als Fileserver läuft der Mars-NWE sehr stabil, und auch als Printserver gibt es keinerlei Probleme mit den Clients, sofern sie eine Netware-Version für DOS-Clients < 4 benutzen, oder Windows-basiert arbeiten. Die 4.11-Version der DOS-Client-Komponente erzeugt jedoch Schwierigkeiten mit dem Druck-Output.

Ein Problem der Netware-Anbindung ist allerdings auch, dass ein Server, der nach den Clients hochgefahren wird, nicht mehr von Windows9x-Systemen erkannt wird. Diese müssen für eine erfolgreiche Anmeldung erneut gestartet werden.

7.5 Vorteile

Die Vorteile des Mars-NWE wurden bereits im Text angerissen, hier sind sie noch einmal zusammengefasst:

- Unter DOS ist die Netware-Client Software wesentlich sparsamer mit dem Speicherplatz als die Alternative "LanManager".
- Durch das Konzept der Novell-Verbindungen wird im Vergleich zu TCP/IP ein höherer Netto-Datendurchsatz des Netzes und somit eine höhere subjektive Geschwindigkeit erreicht.
- Das Verbindungsorientierte Protokoll SPX/IPX sorgt für eine stabilere Verbindung als die SMB-Variante.
- Durch den Aufbau des Mars-NWE übernimmt dieses Paket Routing-Funktionen, die weder konfiguriert noch gewartet werden müssen. Auch nicht-permanente Interfaces wie PPP-Verbindungen werden dabei unterstützt.

8 Apache (2)

Ein Dienstleistungsbereich, der auch im lokalen Netz immer mehr Bedeutung gewinnt, ist die Bereitstellung von HTML-Seiten oder Diensten, die mittels eines Web-Servers angeboten werden können. Diese Dienste können sowohl statischer als auch dynamischer Natur sein:

- **Statisches HTML**: Der Webserver überträgt bei jeder Anfrage dieselbe, unveränderte HTML-Seite, ohne dass der aufrufende Client Einfluss auf die Gestaltung und den Inhalt der Seite hat. Diese Form eignet sich besonders für reine Dokumentation.
- **Dynamisches HTML**: Über aktive Komponenten (Eingabefelder, Auswahllisten, Schaltknöpfe etc.) wird eine interaktive Kommunikation mit dem Client aufgebaut, so dass z.B. Daten aus einer Datenbank abgefragt und dargestellt werden können.

Der Apache (urspr. „a patchy webserver“) ist ein sehr mächtiger HTTP-Server, der den meisten Linux-Distributionen bereits beigefügt ist. Er unterliegt der *Apache License* (ebenfalls eine OpenSource-Lizenz) und ist somit kostenlos und frei verfügbar. Mittlerweile ist die Version 2 des Servers wesentlich weiter verbreitet, so dass an dieser Stelle vorzugsweise auf diese Version eingegangen wird.

Prinzip des Webservers

8.1 Installation

Die Installation des Apache ist prinzipiell nicht schwer, weil die meisten Linux-Distributionen dieses Software-Paket als Binär-Version mitliefern. So ist es unter SuSE und Red Hat als **RPM-Paket**, unter Debian im **DEB**–Format erhältlich, das bereits auf die meisten Anforderungen zugeschnitten ist. Natürlich kann Apache auch als TAR-Ball von **www.apache.org** heruntergeladen werden um es an die eigenen Bedürfnisse anzupassen und zu übersetzen.

Nach dieser Installation ist auch ein Startskript erstellt worden, das dafür sorgt, dass der Server während des Bootvorganges gestartet wird. Manuell kann dieser Server (unter root-Berechtigung) mit dem folgenden Befehl gestartet werden.

```
/etc/init.d/apache start
```

Zu diesem Zeitpunkt ist bereits eine Startseite abrufbar, so dass sich unter der URL

```
http://localhost
```

die Apache-Startseite (oder etwas ähnliches) ansehen lässt. Bei der Debian Sarge Distribution bietet sich folgendes Bild (Ausschnitt:):

Abbildung 8: Apache2-Testseite

In dieser Form ist der Webserver bereits vollständig benutzbar; lediglich zusätzliche Module, die die Funktionalität des Servers beeinflussen, müssen ggf. nachinstalliert werden. Zu diesen Modulen zählen:

- **libphp4**: Das Modul für die Web-orientierte Programmiersprache PHP4.
- **ssl**: Die Komponente, die verschlüsselte Verbindungen unterstützt. Zur Generierung eines eigenen Zertifikates wird zusätzlich das Paket „openssl“ benötigt.
- **perl**: Für „embedded“ Perl benötigt

Durch die Offenheit des Apache stehen noch sehr viele Module zur Verfügung, die aber an dieser Stelle irrelevant sind.

8.2 Eigene HTML-Seiten anbieten

Um nun die selbsterstellten Seiten auch sichtbar machen zu können, müssen sie lediglich an die richtige Stelle kopiert werden. Diese befindet sich bei unterschiedlichen Distributionen leider an unterschiedlichen Stellen, so bevorzugt z.B. SuSE das Verzeichnis

```
/usr/local/httpd/htdocs/
```

während bei Red Hat die Dateien unter

```
/home/httpd/htdocs/
```

abzulegen sind. Um wirklich sicher zu sein, sollte in der Datei /etc/httpd/httpd.conf die Variable „***DocumentRoot***“ zu Rate gezogen werden. Alle Links auf dem eigenen Server werden relativ zu diesem Verzeichnis angegeben, so dass ein Eintrag

```
<A HREF=/meinedateien/index.html>
```

auf die Datei

```
/usr/local/httpd/htdocs/meinedateien/index.html
```

bzw.

```
/home/httpd/htdocs/meinedateien/index.html
```

verweist. Wenn solche Dateien in die Verzeichnisse kopiert werden, ist dabei zu beachten, dass der Apache unter einem eigenen Account (wwwrun oder www) läuft und somit bei der Rechtevergabe für „others“ zumindest das „r“-Recht, bei Verzeichnissen auch das „x“-Recht vorhan-

den ist, sonst bleiben die Dateien unsichtbar. Außerdem muss sichergestellt werden, dass die Dateien grundsätzlich die Endung „.html“ (**klein** geschrieben) erhalten, sonst werden sie nicht als solche angezeigt. Die Datei mit dem Namen „index.html“ bekommt dabei eine besondere Bedeutung: Sie wird angezeigt, wenn die vom Browser übermittelte URL nur den Verzeichnisnamen enthält. Somit liefern die Links

```
http://www.meinserver.de/meinedateien
```

und

```
http://www.meinserver.de/meinedateien/index.html
```

dasselbe Ergebnis, vorausgesetzt, die Datei „index.html“ existiert. Dieses Verhalten kann durch die Variable ***„DirectoryIndex“*** gesteuert werden, so dass z.B. mit dem zusätzlichen Eintrag

```
DirectoryIndex = default.htm
```

auch eine Datei ***default.htm*** als Index verwendet wird. Die Priorisierung ergibt sich dabei aus der Reihenfolge der Einträge.

8.3 Common Gateway Interface (CGI)

Nun existieren aber viele Anforderungen an eine Web-Präsenz, die nicht nur statische Dokumente, sondern auch dynamisch generierte Seiten erfordern. Zu diesem Zweck können beliebige Programme hinterlegt werden, deren einzige Aufgabe es ist, HTML-Code „on the fly“ zu produzieren. Aus der Sicht des Benutzers am Browser erscheinen diese Skripten genauso wie andere HTML-Seiten, nur dass sie je nach Kontext andere Inhalte besitzen.

Der Name dieser Funktion mag ein bisschen verwirren, dann im Laufe der Zeit hat sich die Benutzung dieses Moduls stark geändert. Das CGI war ursprünglich dafür gedacht, daten aus externen Programmen durch den Apache-Server zu veröffentlichen. Jetzt werden extra für diesen Zweck geschriebene Programme eingesetzt, um dynamisch aufgebaute HTML-Seiten zu erzeugen.

In SuSE-Systemen befinden sich die CGI-Programme unter

```
/srv/www/cgi-bin/
```

während Debian diese Dateien unter

```
/usr/lib/cgi-bin/
```

ablegt. Der Nachteil dieser Struktur ist offensichtlich: die Zusammengehörigkeit zwischen den HTML- und den CGI-Dateien für dieselbe Anwendung gehen dabei verloren, so dass die Pflege dieser Seiten erschwert wird. Der Vorteil von CGI-Skripten ist jedoch, dass die Programmierung unabhängig von den Fähigkeiten des Apache sein kann, so dass keine Änderung des Servers nötig ist, wenn z.B. von Perl auf Python gewechselt wird.

So könnte ein Shell-Skript mit folgendem Inhalt in dem jeweils festgelegten CGI-Verzeichnis abgelegt und mittels „*chmod*“-Befehl ausführbar gemacht werden:

```
#!/bin/bash
echo "Content-type: text/html"
echo ""
echo "<html><u1>CGI Funktioniert!</u1>"
echo "<br>Datum:"
date
echo "</html>"
```

Die Ausgabe dieses Skriptes sähe folgendermaßen aus:

```
Content-type: text/html

```

```
<html><u1>CGI Funktioniert!</u1>
<br>Datum:
Mo 29. Sep 16:43:05 CEST 2008
</html>
```

und würde im Browser die entsprechende Ausgabe erzeugen:

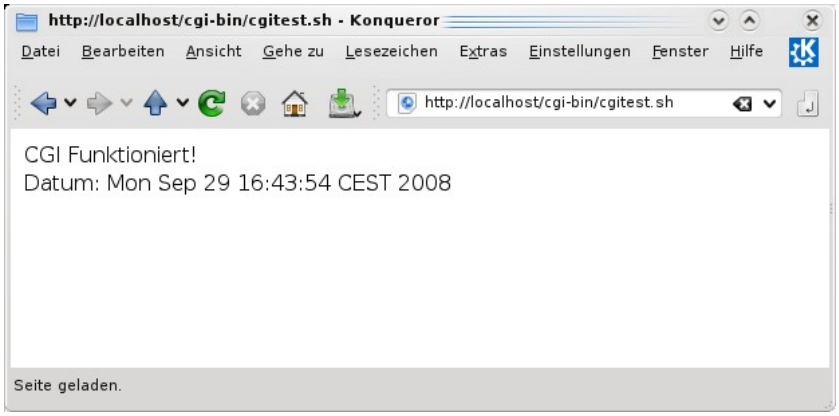


Abbildung 9: Ausgabe eines CGI-Programmes

Als Alternative zu den externen CGI-Programmen sind die im Apache eingebauten („embedded“) Programmiersprachen, wie z.B. PHP.

8.4 Konfiguration

Die gesamte Konfiguration des Apache1 wird über die Datei ***“/etc/httpd/httpd.conf“*** sowie beim Apache2 über ***“/etc/apache2/apache2.conf“*** gesteuert, die – wie die meisten Konfigurationsdateien – aus reinem ASCII-Text bestehen, so dass ein einfacher Texteditor (vi, pico, emacs etc.) ausreicht, um das System zu steuern. Seit Apache2 werden die Informationen in mehrere Dateien aufgespalten, um eine größere Übersichtlichkeit zu erhalten. Ausgehend von der *apache2.conf* wird mittels Include-Direktiven auf die jeweiligen Dateien verwiesen, so dass unterschiedliche Distributionen zwar unterschiedlich lautende Verzeichnisnamen eingerichtet haben, aber alle über die zentrale Konfigurationsdatei ermittelt werden können.

Obwohl die Parameter jetzt in anderen Dateien untergebracht werden, haben sie doch größtenteils ihre Bedeutung behalten, so dass der Umlernprozess von Apache1 nach Apache2 recht gering bleibt.

Besonders hilfreich ist in allerdings die Einrichtung dedizierter Dateien für unterschiedliche „Virtuelle Hosts“, so dass für jede logische Website eine eindeutige Konfigurationsdatei verwendet wird. Allerdings sind die Verzeichnisse durchaus je Distribution unterschiedlich:

SuSE: /etc/apache2/vhosts

Debian: /etc/apache2/sites-enabled

Innerhalb dieser Dateien befinden sich die wichtigen Parameter, auf die im Folgenden eingegangen wird. Da diese Datei allerdings sehr viele Parameter anbietet, seien aus Platzgründen nur die wichtigsten hier beschrieben. Der Bereich eines logischen Servers wird mit dem Tag „***<VirtualHost>***“ eingeleitet und enthält unter anderem diese Einträge:

Variable	Möglicher Wert	Bedeutung
ServerName	www.raum22.de	Unter diesem Namen gibt sich der Server zu erkennen. **Achtung:** Der Zuständige DNS-Server muss diesen Eintrag enthal-

Variable	Möglicher Wert	Bedeutung
		ten!
DirectoryIndex	index.html	Voreingestellte Index-Datei(en) eines Verzeichnisses
DocumentRoot	"/usr/local/httpd/htdocs"	Dieses ist der Pfad, in dem der Server alle HTML-Dateien erwartet.

In dieser Sektion können weiterhin andere Verzeichnisse eingebunden werden, die nicht unterhalb des ***DocumentRoot*** liegen. So wird verhindert, dass jemand mithilfe der manuell eingetippten URL trotz eingeschaltetem Schutz die CGI-Skripten einsehen kann. Daher wird das Verzeichnis cgi-bin auf diese Weise eingebunden:

```
Alias /cgi-bin/ "/usr/local/httpd/cgi-bin/"
```

Weiterhin können bestimmte Optionen für einzelne Verzeichnisse angegeben werden, die das Verhalten des Servers für diese beeinflussen. So wird auch das CGI-Verzeichnis auf diese Weise konfiguriert:

```
<Location /cgi-bin>
AllowOverride None
Options +ExecCGI -Includes
SetHandler cgi-script
</Location>
```

Die Option ***„+ExecCGI“*** erlaubt dem Server, die darin enthaltenen Programme auszuführen. Selbst wenn in diesem Verzeichnis eine gültige HTML-Datei existiert, kann sie nicht als solche angezeigt werden.

Ein weiterer häufig verwendeter Eintrag sind die Aliase für Benutzerverzeichnisse. Über diesen Eintrag können alle eingerichteten Linux-Benutzer ihre eigenen HTML-Seiten publizieren:

```
UserDir public_html
```

Dieser Eintrag gibt den Verzeichnisnamen vor, den Benutzer in ihrem eigenen Home-Directory für die Publizierung ihrer Seiten verwenden müssen. Alle anderen Verzeichnisse bleiben unberücksichtigt. In der URL wird dieses Verzeichnis mit dem Benutzernamen substituiert und mit einer Tilde „~“ versehen. Wenn z.B. der Benutzer „hugo“ in seinem Home-Directory */home/hugo* ein Verzeichnis *public_html* erzeugt, ist es via HTTP unter der URL „*http://www.raum22.de/~hugo*“ zu erreichen. Auch hier muss darauf geachtet werden, dass der Apache-User („www“ oder „wwwrun“) Lese-Rechte besitzt. Soll diese Eigenschaft gesperrt werden, muss der Eintrag in

```
UserDir disabled
```

geändert werden.

8.5 Das PHP-Modul

Nur statische HTML-Seiten auszuliefern ist nicht nur langweilig, sondern auch meistens nicht ausreichend für die Aufgaben, die ein Web-Server erfüllen soll. CGI-Skripten können zwar dynamische Inhalte erzeugen, statische Bereiche liegen jedoch getrennt von ihnen.

Diese Trennung kann beispielsweise mit der Skriptsprache PHP überwunden werden. Diese wird durch ein Modul dem Apache-Server hinzugefügt, wodurch HTML-Dateien direkt mit PHP ergänzt und so die Programmierung vereinfacht werden kann. Um den PHP-Interpreter als Bestandteil des Apache-Servers zu integrieren, muss das entsprechende Modul (*mod_php5*) beim Start des Apache-Servers geladen werden. Unter SuSE geschieht das durch einen Eintrag

in der Datei

```
/etc/sysconfig/apache2
```

Dort wird die Umgebungsvariable „**APACHE_MODULES**“, in der die Option „**php5**“ hinzugefügt werden muss. Ohne diesen Eintrag ist das Modul zwar im Betriebssystem vorhanden, aber nicht im Apache aktiviert. In Debian-basierten Systemen geschieht die Aktivierung auf anderem Wege. Im Apache-Verzeichnis befinden sich die beiden Unterverzeichnisse „mods-available“ und „mods-enabled“. Wie der Name schon andeutet, werden die Konfigurations-Teile der httpd.conf in dem Verzeichnis „mods-available“ komplett abgelegt. Soll ein Modul auch verwendet werden, müssen die entsprechenden Dateien und das Verzeichnis „mods-enabled“ kopiert – oder besser gelinkt – werden. Im Fall von PHP5 geschieht das durch den Befehl

```
ln -s /etc/apache2/mods-available/php5.* /etc/apache2/mods-enabled/
```

Nach einem Neustart des Apache sollte das System PHP-Dateien erkennen und ausführen. Der einfachste Test besteht darin eine Datei „***testphp.php***“ zu erzeugen, die folgende Zeilen enthält:

```
<?php
     phpinfo();
?>
```

Sofern diese Datei innerhalb des „DocumentRoot“-Verzeichnisses abgelegt wird, kann diese dann unter der URL

```
http://localhost/testphp.php
```

angesehen werden. Als Ausgabe sollte dabei eine Übersicht über die vorhandene Apache- und PHP-Umgebung angezeigt werden:

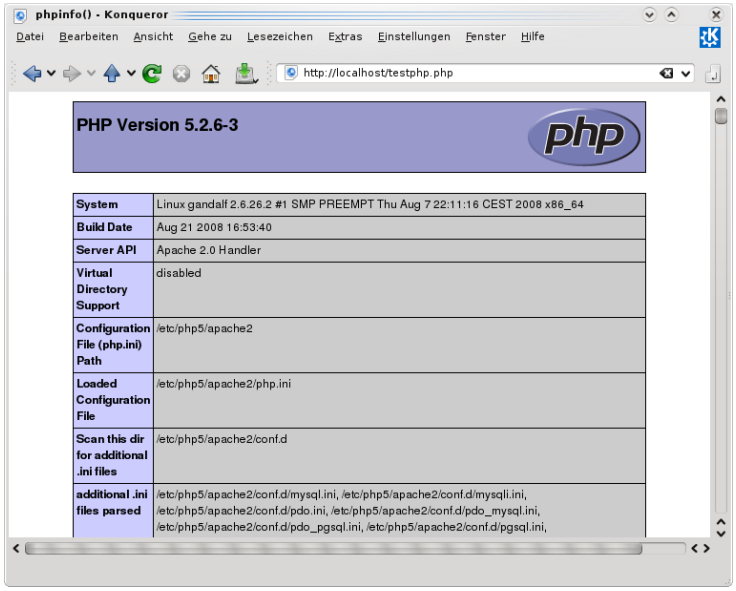


Abbildung 10: Ausgabe von phpinfo()

Achtung: Da die Ausgabe von phpinfo() sehr umfangreich ist, eignet sie sich hervorragend als Informationsquelle für einen Systemeinbruch. Bitte entfernen Sie diese Testdatei umgehend wieder aus dem System!

8.6 Der „sichere“ Apache

Der Webserver bietet mehrere Möglichkeiten den Zugriff zu sichern, wobei die unterschiedliche Ziele verfolgt werden. Die erste Möglichkeit besteht darin, nur bestimmte Rechner an die Website heranzulassen. Zusätzlich können noch Apache-spezifische User angelegt werden, die vor dem Zugriff auf den Webserver mit einem Anmeldedialog abgefragt werden können.

Beide Dieser Sicherungsformen können entweder zentral für den gesamten Server oder für jedes einzelne Verzeichnis (incl. Unterverzeichnisse) gesetzt werden. Zu letzterem Zweck muss in der zentralen *httpd.conf* die Variablen

```
AllowOverride All
AccessFileName.htaccess
```

gesetzt werden. Für jedes Verzeichnis wird dann automatisch der Inhalt der Datei *.htaccess* in eben diesem Verzeichnis ausgewertet, in dem die notwendigen Eigenschaften einstellen zu können.

Soll beispielsweise nur der PC mit der IP-Adresse 192.168.1.20 Zugriff erhalten, muss der Inhalt der Datei wie folgt aussehen:

```
order deny, allow
deny from all
allow from 192.168.1.20
```

Analog zu den Netzadressen kann ebenso ein Bereich mittels der Netzmaske definiert werden. Die letzte Zeile müsste dann so aussehen:

```
allow from 192.168.1.0/255.255.255.0
```

Wenn allerdings eine Benutzerabfrage über die Zugangsberechtigung entscheiden soll, müssen zunächst diese Benutzer definiert und eingerichtet werden. Dieses geschieht über die Datei, die in dem Eintrag *AuthUserFile* angegeben wurde, wie in folgendem Beispiel:

```
AuthUserFile  /etc/httpd/passwd
```

Diese Passwortdatei hat wenig mit der zentralen Benutzerverwaltungsdatei /etc/passwd gemein, sondern muss zusammen mit dem ersten Benutzer über den Befehl

```
htpasswd –c /etc/httpd/passwd <benutzername>
```

angelegt werden.

Jedes Verzeichnis, das nun vor unberechtigtem Zugriff geschützt werden soll, wird nun um die Datei ***.htaccess*** ergänzt, die folgenden Inhalt aufweisen muss:

```
AuthType basic
AuthName "Geschuetzter Bereich"
require valid-user
```

Nun sorgt der Apache dafür, dass nur diejenigen Clients eine Verbindung bekommen, die auch einen in der Passwort-Datei abgelegten Benutzer vorweisen können.

8.7 Secure Socket Layer (SSL)

Besonders interessant ist jedoch die Möglichkeit, eine verschlüsselte Verbindung zwischen Client und Server aufbauen zu können, wenn z.B. eine Angabe von persönlichen Daten nicht im Klartext über das Internet gesendet werden soll. Voraussetzung hierzu sind einerseits das Paket „mod_ssl“, um den Apache überhaupt mit den notwendigen Funktionen auszustatten, andererseits Das Paket „openssl“, um die Erzeugung und Verwaltung der Schlüssel sowie die Ver- und Entschlüsselung der Daten selbst zu ermöglichen. Analog zu dem PHP-Modul muss unter SuSE die Konfigurationsdatei ***„/etc/sysconfig/apache2“*** angepasst und unter Debian die entsprechnden Dateien in das „**mods-enabled**“- Verzeichnis verlinkt werden.

Zunächst muss ein **privater Schlüssel** erzeugt werden, der vor der Öffentlichkeit geschützt werden muss. Dieses geschieht mit dem Befehl

```
openssl genrsa –out meinserver.key
```

Anschließend muss ein **Certificate Signing Request** (CSR) generiert werden, in dem die Angaben über den Server und den zuständigen Administrator abgelegt werden müssen. Der hierzu nötige Befehl lautet:

```
openssl req -new -key meinserver.key -out meinserver.csr
```

Die Generierung des CSR erfordert einige Angaben über den zukünftigen Webserver, die auch dem Client mitgeteilt werden, wenn das Zertifikat anerkannt werden soll. Diese Fragen sollten also wahrheitsgemäß und möglichst vollständig beantwortet werden. Die Frage nach dem „Namen“ ist dabei leider etwas missverständlich: Es wird nicht der Name des Administrators, sondern der FQDN des späteren Webservers (z.B. „www.meinserver.de“) erwartet.

Der letzte Schritt ist die Erzeugung eines öffentlichen Zertifikates, das dem Client über geben und von ihm anerkannt werden muss. Im produktiven Umfeld kann dieses über eine öffentliche Zertifizierungsstelle (Certification Authority, „CA“) geschehen, um dem Client die Sicherheit geben zu können, dass der Absender auch wirklich derjenige ist, der im Zertifikat enthalten ist.

Da eine solche Zertifizierung Geld kostet, kann für lokale Zwecke auch ein eigenes Zertifikat erstellt werden:

```
openssl x509 -req -days 365 -in meinserver.csr -signkey \
meinserver.key -out meinserver.crt
```

Das entstehende Zertifikat wird entweder per Diskette übertragen, oder aber online vom Client anerkannt, was natürlich ein Sicherheitsrisiko birgt. Sofern jedoch nur ein lokales Netz oder nur eine begrenzte Anwendergruppe betroffen ist, reicht diese Form der Zertifizierung völlig aus.

Damit nun der Apache auch die SSL-Verbindungen aufbauen kann, müssen die Zertifikate bekannt gegeben und die SSL-Verschlüsselung aktiviert werden. Auch hier werden wiederum Variablen in der jeweiligen Modul-Konfigurationsdatei benötigt. Während Debian die entsprechenden Links im Verzeichnis */etc/apache2/mods-enabled* benötigt, wird bei SuSE dieses über eine Umgebungsvariable gesteuert, die in der Datei */etc/sysconfig/apache2 abgelegt wird (*APACHE_SERVER_FLAGS="**SSL**")

SSL-Verbindungen werden im Gegensatz zu „normalen“ http-Verbindungen nicht über Port 80/tcp sondern über Port 443/tcp aufgebaut werden, muss unter Debian die */etc/apache/ports.conf* ergänzt werden, so dass sie die folgenden Zeilen enthält:

```
Listen 80
Listen 443
```

Nun kann der SSL-Service direkt konfiguriert werden, wozu ein neuer Virtueller Host erzeugt wird. Dazu wird eine neue Datei im entsprechen Verzeichnis angelegt, in dem die SSL-Engine aktiviert und mit den darauf folgenden Einträgen die Schlüssel und Zertifikate bekannt gegeben werden. Durch die Angabe „:443“ im Namen des virtuellen Hosts wird festgelegt, dass dieser ausschließlich für Verbindungen über Port 443/tcp aktiviert wird. Der Beginn einer solchen Datei könnte dann folgendermaßen aussehen:

```
NameVirtualHost *:443
<VirtualHost *:443>
        ServerAdmin dv-verwaltung@uni-bielefeld.de
        SSLEngine on
        SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDI-
UM:+LOW:+SSLv2:+EXP:+eNULL
        SSLCertificateFile /etc/apache2/ssl/meinserver.crt
        SSLCertificateKeyFile /etc/apache2/ssl/meinserver.key
```

Es muss natürlich vorausgesetzt werden, dass die angegebenen Pfade zu den Zertifikaten korrekt und für den Apache lesbar sind.

Wenn nun der Apache neu gestartet wird, müsste bereits eine SSL-Verbindung mit einem beliebigen Browser möglich sein.

9 Datenbanken

9.1 MySQL

Eine der häufigsten Datenbanken, die in Verbindung mit Linux installiert werden, ist MySQL. Sie wurde entwickelt, um eine möglichst hohe Geschwindigkeit zu erzielen und dabei auch noch möglichst wenig Ressourcen zu verwenden. Daraus ergibt sich, dass zunächst alle nicht unbedingt notwendigen Features außer Acht gelassen und erst nachträglich hinzu gefügt wurden. Daher wird sie von „eingefleischten“ Datenbank-Admins nicht wirklich ernst genommen, da ihr nur mit einigem Aufwand das abverlangt werden kann, was integraler Bestandteil eines Relationalen Datenbank-Systems sein sollte.

Trotzdem hat sich MySQL einen sehr hohen Rang in der Liste der Web-Datenbanken erkämpft, zumal fast alle Web-Applikationen diese Datenbank unterstützen. In den meisten Fällen wird bei der Installation dieser Applikationen lediglich vom Benutzer verlangt, die Datenbank-Software zu installieren, eine Datenbank und einen entsprechenden Benutzer für dessen Zugriff einzurichten. Da nun MySQL ohnehin **sehr** umfangreich ist, will ich mich auch nur auf die genannte Aufgabe beschränken. Zu tiefer gehenden Informationen ist die einschlägige Literatur besser geeignet. Auch bei MySQL (=SUN) selbst gibt es viel Informationen, doe kostenfrei verfügbat sind. Beispiel: **http://dev.mysql.com/doc/refman/5.1/de/**

9.1.1 Installation

Der Datenbank-Server ist üblicherweise schnell installiert, indem das entsprechende Paket direkt aus der Distribution oder von **www.mysql.org** heruntergeladen und eingerichtet wird.

Unter SuSE-Systemen geschieht das durch den Yast entweder interaktiv oder duch die Auswahl der beiden Pakete „mysql“ und „mysql-administrator“

```
zypper install mysql mysql-administrator
```

In Debian-basierten Systemen wird analog dazu das Werkzeug „apt-get“ verwendet:

```
apt-get install mysql-server mysql-client
```

Der „mysql-administrator“ ist zwar nicht im Standard-Repository, ist aber über die mysql-Website zu bekommen.

9.1.2 Einrichtung von Datenbanken und Benutzern

Nachdem dieser durch das zugehörige init-Skript gestartet wurde, steht genau eine Datenbank mit dem Namen „mysql“ und ein einziger Benutzer „root“ (ohne Passwort!!) zur Verfügung, um weitere Objekte einzurichten. Der einfachste Weg, mit dieser Datenbank zu arbeiten, führt über die Shell, und dem Tool „mysql“:

```
burkhard@gandalf:~$ mysql -u root mysql
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 71
Server version: 5.0.51a-15 (Debian)
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> _
```

Nun ist es recht unklug, alle Daten in dieser einen Datenbank abzulegen, schließlich gibt es auch im Filesystem Verzeichnisse. Also besteht die erste Handlung darin, die erste eigene Datenbank anzulegen und sich dann mit dieser zu verbinden:

```
mysql> create database meinedb;
Query OK, 1 row affected (0.00 sec)

mysql> use meinedb;
Database changed
mysql> _
```

Hier wird also eine Datenbank angelegt, die den Namen „meinedb" trägt. Der „use"-Befehl verändert den „Standort", alle danach eingegebenen Befehle werden automatisch auf Objekte in dieser neuen Datenbank bezogen. Da es auch nicht sinnvoll ist, immer mit dem „root" angemeldet zu sein, soll ein neuer Benutzer angelegt werden, der auch sämtliche Berechtigungen auf diese neue Datenbank bekommen soll. In diesem Beispiel ist es der Benutzer „hugo" mit dem Passwort „Demo123". Da dieser Benutzer sich auf demselben Rechner wie die Datenbank befindet, wird der Datenbank der Name des Rechners „localhost" mitgeliefert. Diese Verknüpfung muss bei der Generierung des Benutzer-Logins als „@localhost" mit angegeben werden::

```
mysql> create user hugo@localhost identified by 'Demo123';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all on meinedb.* to hugo@localhost;
Query OK, 0 rows affected (0.00 sec)
```

Dieser Zusatz verhindert auch, dass eine Verbindung von „außerhalb", also von einem anderen Rechner zugelassen wird. Sollte also eine Verbindung zur MySQL-Datenbank von dem Rechner „PC03" möglich sein, so muss der Benutzer „**hugo@pc01**" entsprechend autorisiert werden. Sollte die Datenbank von jedem Rechner aus ansprechbar, kann statt des Rechnernamens ein Prozentzeichen „%" verwendet werden. Dieses gilt in der SQL-Syntax als Wildcard und passt daher auf jeden Rechnernamen außer dem „localhost".

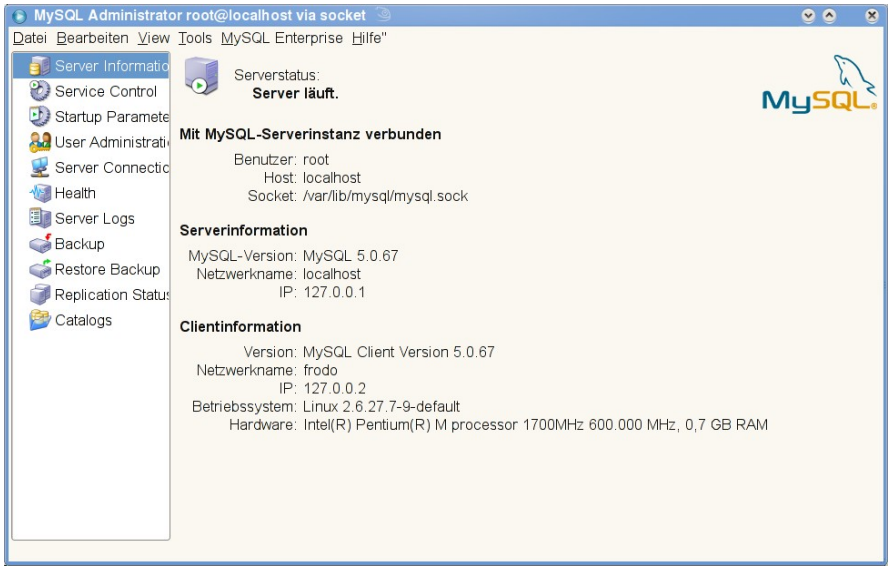
Sollte im Nachhinein das Passwort des Benutzers geändert werden, geschieht dieses durch den „set password"-Befehl:

```
set password for hugo@localhost=PASSWORD('123Demo');
Query OK, 0 rows affected (0.00 sec)
```

Die Sprache, mit der die Datenbank bearbeitet wird ist glücklicherweise die standardisierte „Structured Query Language" (=SQL), so dass die meisten Befehle in allgemein gültigen SQL-Nachschlagewerken nachzulesen sind. Die MySQL-Spezifischen Ergänzungen können allerdings nur in der MySQL-Dokumentation nachgelesen werden, die unter der Adresse **http://dev.mysql.com/doc/refman/5.1/en/** zu finden ist.

9.1.3 Grafische-Tools

Haben Sie die Möglichkeit, die X-Oberfläche nutzen zu können, brauchen Sie nicht die SQL-Syntax zu lernen, um Ihre Datenbank zu administrieren. Das Paket „mysql-administrator" bietet eine komfortable Möglichkeit, auf Ihre Datenbank zuzugreifen. Diese muss nicht zwangsweise auf dem eigenen Rechner liegen, sondern kann auch aus der Ferne genutzt werden. Lediglich der Rechnername bzw. Die IP-Adresse und ein gültiges Benutzer/Passwort-Paar muss bekannt sein. Wenn nun nicht noch eine Firewall die Verbindung stört, kann eine Entfernte Datenbank mit diesem Werkzeug genutzt werden.



Über diese Verbindung können sogar Datenbank-Backups auf dem eigenen Rechner abgelegt werden, um beispielsweise die Datenbank auf einen anderen Rechner zu übertragen.

Aus technischer Sicht wird von MySQL das Dateisystem als Basis für die jeweiligen Tabellen verwendet. Das Hauptverzeichnis für Datenbanken ist meistens /var/lib/mysql und enthält die zentralen Einstellungen des MySQL-Servers. Jede angelegte Datenbank wird duch ein gleichnamiges Unterverzeichnis abgebildet, das wiederum die für jede Tabelle 2 Dateien anlegt: Eine für die eigentlichen Daten (*.MYD), eine für Indexe (*.MDI) und eine weitere für zusätzliche Informationen. Sofern die Datenbank gestoppt ist, reicht es aus, dieses Verzeichnis zu sichern, um eine Offline-Kopie sicherzustellen.

Ein solchermaßen gesichertes Verzeichnis kann auf einer anderen Betriebssystemplattform direkt eingespielt werden, um eine Datenbank zu klonen.

MySQL ist die Basis für viele andere Produkte. Zu diesen zählen prominante Anwendungen wie das CMS „Joomla!“ oder „Drupal“, Groupware-Systeme wie „eGroupware“ oder „PHProject“ aber auch komplexte ERP-Systeme wie „vTiger“. Da der Aufbau von MySQL sehr kompakt und übersichtlich ist, kann es mit geringem Aufwand auf fast jede beliebige Plattform portiert werden, was auch bereits geschehen ist. MySQL gibt es zurzeit nicht nur für Linux und Windows, sondern auch für HP-UX, Solaris, AIX und noch viele mehr, weil der Source Code verfügbar ist und mit vergleichsweise geringem Aufwand an die jeweilige Plattform angepasst werden kann.

10 Firewall mit Linux

Es gibt mittlerweile eine große Menge an Firewall-Tools, die fast alle nach demselben Prinzip arbeiten: Während die Kernel-Routingfunktionen ausgeschaltet werden, übernimmt das Firewall-Tool (bzw. *die* Firewall-Tool*s*, wenn es sich um mehrere Module eines Produktes handelt) die Kontrolle über die übertragenen Datenpakete. Dieses Verfahren ist zwar sehr sicher, besitzt aber einen Nachteil, dass diese Komponenten vor ihrem Einsatz auf das Betriebssystem abgestimmt und kompiliert werden müssen.

Weil nun schon der Kernel selbst umfangreiche Möglichkeiten bietet und außerdem die komplette Beschreibung der verfügbaren externen Module den Umfang dieses Kapitels sprengen würde, werden nur die Kernel-Funktionen sowie deren Konfiguration beschrieben.

10.1 Benötigte Software

Das ursprüngliche Produkt „**ipfwadm**“ (IP-Firewall-Administration) war der erste Einstieg in die Firewall-Konfiguration und wurde mit den Kernel-Versionen 2.1.x und 2.2.x durch „**ipchains**“. ersetzt. Seit Kernel 2.3.x ind 2.4.x wurden beide durch das **netfilter**-Paket abgelöst, mit dem ein neues Konzept in den Kernel eingebaut wurde. Die jüngsten Informationen und die verfügbaren Downloadsites sind unter dieser Adresse zu erreichen:

http://netfilter.filewatcher.org/ (Kernel 2.4.x).

All diese Produkte stellen nur ein Administrationstool für die im Kernel vorhandenen Filter- und Logging- und Masquerading-Funktionen zur Verfügung, also keinerlei logische Überprüfung der Firewall-Regeln. Die Benutzung dieser Tools ist nicht sonderlich schwer, die Schwierigkeit liegt – wie so oft – im Detail. Die richtige Zusammensetzung der Firewall-Regeln ist das vorrangige Problem. Außerdem birgt eine Firewall-Administration über ein Netzwerk hinweg immer die Gefahr, dass durch einen unachtsam abgesetzten Befehl die eigene Wegstrecke gekappt und die Bedienung des Firewalls unmöglich wird. Daher sollte bei jeder Veränderung sichergestellt sein, dass der Rechner auch manuell erreichbar ist.

Die Tools ipfwadm, ipchains wie auch iptables werden relativ ähnlich gehandhabt, doch bietet iptables zusammen mit dem Kernel 2.4.x – in dem der komplette TCP/IP-Stack zum wiederholten Mal überarbeitet wurde – sehr viele neue Möglichkeiten der Optimierung und Konfiguration. Aufgrund dessen wird sich der folgende Text ausschließlich auf die jüngere Version (Kernel 2.4.x) beziehen.

Im Kernel müssen diese Fähigkeiten natürlich eingebunden werden, weil sonst die beschriebenen Tools nicht angewendet werden können. Konkret müssen die folgenden Variablen in den Networking options gesetzt sein:

```
[*] Network packet filtering (replaces ipchains)
[ ] Network packet filtering debugging
[*] Socket Filtering
```

Als Untergruppierung ist seit dem 2.4.0 die Gruppe „Netfilter Configuration“ hinzugekommen. In dieser sollten die folgenden Eigenschaften hinzugefügt werden:

```
<M> Connection tracking (required for masq/NAT)
<M> FTP protocol support
<M> Userspace queueing via NETLINK (EXPERIMENTAL)
<M> IP tables support (required for filtering/masq/NAT)
<M>  limit match support
<M>   MAC address match support
<M>   netfilter MARK match support
```

```
<M>   Multiple port match support
<M>   TOS match support
< >   tcpmss match support
<M>   Connection state match support
<M>   Unclean match support (EXPERIMENTAL)
<M>   Owner match support (EXPERIMENTAL)
<M>   Packet filtering
<M>     REJECT target support
<M>     MIRROR target support (EXPERIMENTAL)
<M>   Full NAT
<M>     MASQUERADE target support
<M>     REDIRECT target support
<M>   Packet mangling
<M>     TOS target support
<M>     MARK target support
<M>   LOG target support
< >   TCPMSS target support
```

Natürlich können auch alle Pakete fest in den Kernel gelinkt werden, doch benötigen sie dermaßen viel Ressourcen, dass es sinnvoller ist, nur die Module zu laden, die auch wirklich benötigt werden. Daher ist die Auswahl als Modul (für diesen Zweck) die geeignetere Wahl

10.2 Konfiguration

„iptables"-Syntax:

```
iptables -[ADC] chain Regel [Optionen]
iptables -[RI] chain RegelNr. Regel [Optionen]
iptables -D chain RegelNr. [Optionen]
iptables -[LFZ] [chain] [Optionen]
iptables -[NX] chain
iptables -P chain target [Optionen]
iptables -E alter-chain-Name Neuer-chain-Name
```

Um einen Firewall zu konfigurieren, wird eine Folge von ***„iptables"***-Aufrufen verwendet, in dem jeweils eine Firewall-Regel in eine vorgegebene Kette von Regeln eingefügt wird. Mehrere dieser Ketten steuern zusammen den Filtermechanismus.

Jede Regel setzt sich aus einer chain, Attribute des Paketes und einer Aktion zusammen, wobei die „chains" Ketten von Regeln darstellen, die jeweils für eine bestimmte Position im Filtersystem vorgesehen sind. Hierzu werden 5 verschiedene Typen von Regeln definiert, die sog. „**chains**":

- **PREROUTING**: Alles, was in den Firewall hineinkommt, noch bevor eine Routing-Entscheidung getroffen wurde. Hierzu zählen auch Pakete, die über das loopback-Device gesendet werden.
- **INPUT**: Alle Daten, die für interne Prozesse des Firewalls bestimmt sind
- **OUTPUT**: Alles, was den Firewall von einem internen Prozess verlässt.
- **FORWARD**: Alles, was von dem Firewall mittels-Routing-Funktionen durchgeschleust wird.
- **POSTROUTING**: Alle Datenpakete, die den Firewall verlassen, also sowohl von internen Prozessen als auch vom Forwarding

Diese chains gelten als vordefinierte Regel-Gruppen mit einer oder mehrerer Regeln, die jeweils sequenziell abgearbeitet werden. Es können für komplexere Definitionen eigene chains eingesetzt werden, die dann als Substitution für ein Target in dem Regelwerk wiederverwendet werden können. Welche Aktionen allerdings in welcher Kette zulässig sind, wird über die **Tables** gesteuert, die alle möglichen Aktionen zu sinnvollen Gruppen zusammenfasst.

Diese Form der Zusammenfassung von Aktionen ist übrigens ein neues Feature, das mit der „Modernisierung“ des Routing- und Firewalling-Stacks des Kernels hinzugekommen ist.

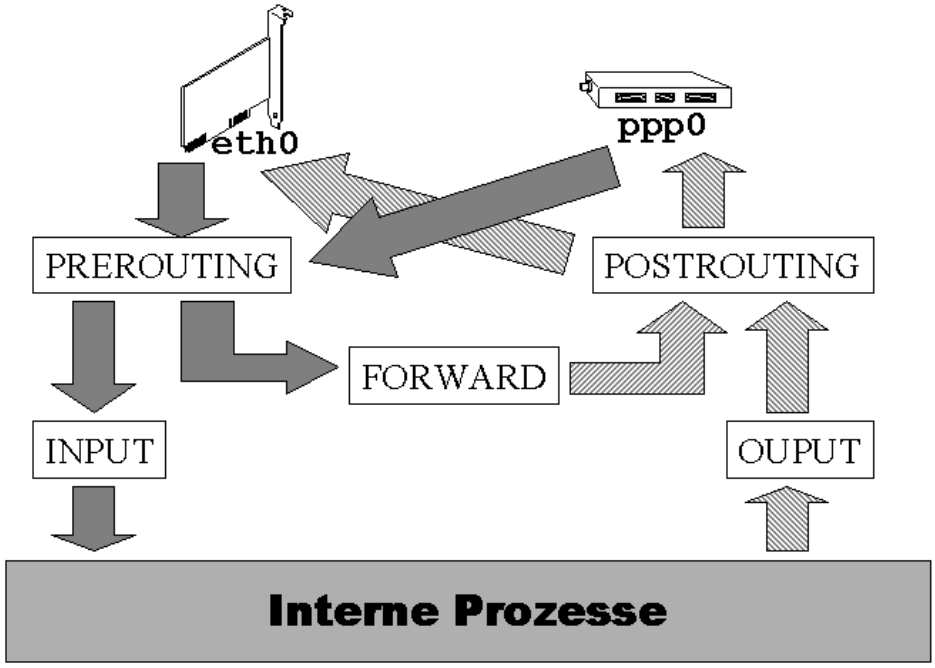


Abbildung 11: Prinzip der IPTables

Das obige Bild beschreibt das Prinzip des Filtersystems. Jedes Datenpaket, das direkt von einem Prozess des Rechners ausgeht oder an ihn adressiert ist, passiert jeweils nur den Input- oder den Output-Filter. Ein Paket, das für das Forwarding vorgesehen ist, durchläuft nicht mehr alle drei, sondern nur noch den Forward-Filter.

Jede chain ist mit einer „Policy“ belegt, die beschreibt, was mit einem Datenpaket geschieht, für das keine der angegebenen Regeln greift. Eine Policy besteht immer aus einem „Target“ (s.u.), so dass entweder alles erlaubt ist, was nicht ausdrücklich verboten, oder alles verboten, was nicht ausdrücklich erlaubt ist. In den meisten Fällen wird die letztere benutzt, weil sie weitestgehend verhindert, dass das betroffene System „aus versehen“ völlig offen ist.

Bei der sequenziellen Abarbeitung führt die erste zutreffende Regel dazu, dass sofort in das „Target“ (s.u.) gesprungen wird, sofern eines angegeben wurde. Weil alle folgenden Regeln dabei ignoriert werden, ist es wichtig, auf die Reihenfolge der Regeln zu achten.

Zur Manipulation dieser chains werden Regeln hinzugefügt, gelöscht oder verändert, was mit diesen Optionen geschieht:

- **–A <chain> <Regel>:** Regel wird an „Chain“ angehängt.
- **–D <chain> <Nummer>:** Regel mit Nummer „nummer“ wird gelöscht
- **–R <chain> <Nummer> <Regel>:** Regel mit Nummer „nummer“ wird gegen die angegebene ersetzt.
- **–I <chain> <Nummer> <Regel>:** Regel wird vor der Regel mit der Nummer <Nummer> eingefügt.
- **–F <chain>:** Alle Regeln in „chain“ werden gelöscht.
- **–P <chain> <target>:** „chain“ wird mit „Target“ voreingestellt (Policy).

Weiterhin besitzt jede Regel ein Ziel, ein „**Target**“, das beschreibt, was mit dem Paket geschehen soll, wenn die betreffende Regel angewandt werden kann. Die Targets werden von unterschiedlichen „Tables“ angeboten und benötigen daher ggf. zusätzliche Module :

- **ACCEPT:** Das Paket wird hindurchgelassen (table: filter)
- **REJECT:** Das Paket wird gestoppt und an den Sender eine ICMP-Meldung geschickt. (table: filter)
- **DROP:** Das Paket wird ohne weitere Aktion gestoppt (Keine Rückmeldung) (table: filter)
- **MASQUERADE:** Das Paket wird hindurchgelassen, aber dabei maskiert (nur für POST-ROUTING) (table: nat)
- **REDIRECT:** Das Paket wird auf eine lokale Portadresse umgeleitet. Dieses funktioniert nur bei „REROUTINF“-Regeln. (table: nat)
- **RETURN:** bricht die aktuelle Chainauswertung ab und springt in die übergeordnete Chain oder (bei einer Basis-Chain) in das Policy-Target.
- **<Ohne Target>:** Die Überprüfung wird bei der nächsten Regel fortgesetzt. Dieses wird hauptsächlich beim Accounting benutzt.
- **SNAT:** Source NAT: Die Absenderadresse wird verändert. (table:nat)
- **DNAT:** Destination NAT: Die Zieladresse wird verändert. (table:nat)

Um Die Richtung des Datenpaketes angeben zu können, muss natürlich auch Quelle und Ziel bestimmt werden sowie das Interface, das von dieser Regel betroffen ist. Zu diesem Zweck werden folgende Optionen benötigt:

- **-p**: Protokoll, über das kommuniziert wird. Werte: „tcp“, „udp“, „icmp“. Diese Option muss den Portfilterangeben auf jeden Fall vorangestellt werden.
- **-s**: (source) Quelle des Paketes in IP-Schreibweise, z.B. 195.225.0.0/16
- **--sport**: (source port) Port-Quelle des Datenpaketes, z.B. „1024:65534“ oder „53“. Benötigt zwingend die „-p <protokoll>“-Angabe
- **-d**: (destination) Ziel des Paketes in IP-Schreibweise, z.B. 192.168.10.0/24
- **--dport**: (destination port) Port-Ziel des Datenpaketes, z.B. „1:1024“ oder „1526“. Benötigt zwingend die „-p <protokoll>“-Angabe
- **-i**: (in-interface) Gerät, über das das Paket in das System hereinkommt wird, z.B. „eth0“
- **-o**: (out-interface) Gerät, über das das Paket das System verlässt, z.B. „ppp0“
- **! *<option>*** : Negation: „! –d 192.168.8.1“ bezeichnet beispielsweise alle Pakete, die **nicht** an 192.168.8.1 gerichtet sind.

Wird einer von diesen Werten nicht angegeben, impliziert das die Irrelevanz des Parameters. Ist es für eine bestimmte Regel egal, welche Adresse der Empfänger besitzt, so kann statt der Angabe „-d 0/0“ dieser Parameter auch ganz weggelassen werden.

Achtung: Werden Ports angegeben, **muss** ein Protokoll (TCP oder UDP) angegeben werden, weil unter ICMP keine Ports verwendet werden. Diese Option (-p) muss zwingend den anderen Filterangeben **vorangestellt** werden, weil sonst das benötigte Modul nicht geladen werden kann. Irritierenderweise wird in so einem Fall die Fehlermeldung „unknown option“ ausgegeben.

Alle Regeln sollten grundsätzlich mit den IP-Adressen deklariert werden, nicht mit den Rechnernamen, wie sie im DNS abgebildet sind, weil unter Umständen während des Regelaufbaues die DNS-Dienste nicht verfügbar sind und somit alle Regelungen nicht eingefügt werden.

10.3 Beispielkonfigurationen

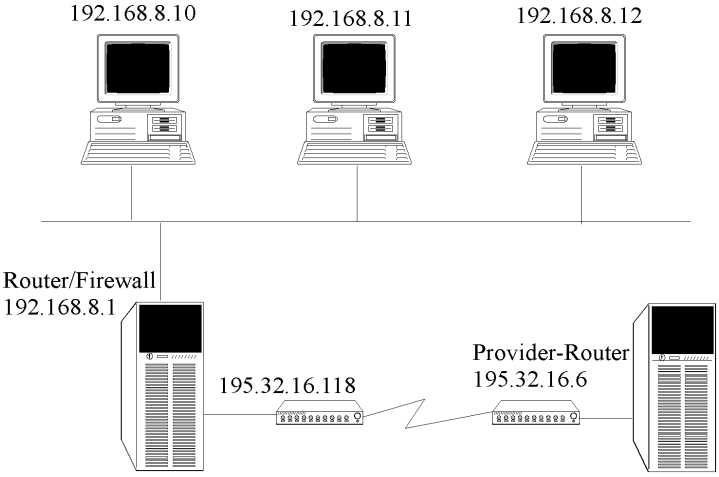


Abbildung 12: Beispiel-Netz für Firewall-Einsatz

Die Adresse des inneren Netzes sei 192.168.8.0/24, der Firewall sei hier der Einfachheit halber auch der default Router, mit der internen Adresse 192.168.8.1 auf dem Gerät eth0 und der externen Adresse 195.32.26.118 auf dem Gerät ppp0, die als statisch vorausgesetzt wird. Der Router des Providers mit der Adresse 195.32.26.1 enthält auch gleichzeitig den Proxy für HTML-Zugriffe, (auf Port 80) und den DNS. Es sollen nun folgende Regeln abgebildet werden:

1. Der Zugriff vom Firewall nach Außen darf nur über die freien auf die reservierten Ports des Internets geschehen. Da das nur das innere Netz betrifft, wird diese Regel nur im FORWARD-Chain benötigt:

```
# Zuerst alle Regeln loeschen
/usr/sbin/iptables -F FORWARD

# Sofern keine Regel greift, wird die Policy
# DROP aktiv
/usr/sbin/iptables -P FORWARD DROP

# Der Zugriff auf den Providerrouter wird
# zugelassen
/usr/sbin/iptables -A FORWARD -p tcp \
 -s 195.32.16.118/32 --sport 1024: \
 --dport 1:1023 -j ACCEPT

# Alle Daten, die zu einer bestehenden
# Verbindung gehoeren, sind OK
/usr/sbin/iptables -A FORWARD -m state \
 --state ESTABLISHED,RELATED -j ACCEPT
```

2. Der Zugriff direkt auf den Firewall soll möglichst vollständig gesperrt werden, von innen sollte jedoch die Möglichkeit der Administration offen bleiben. An dieser Stelle kommt das neue Modul „state“ zum Einsatz, mit dessen Hilfe der logische Zustand ermittelt werden kann. Die 3. Regel der nachfolgenden Liste werden all die Datenpakete zugelassen, die zu einer bereits bestehen Verbindung gehören. Dazu zählen einerseits die „Rückwege“ einer bestehenden TCP-Verbindung (Zustand „*ESTABLISHED“*), andererseits neu aufgebaute TCP-Kanäle, die vom Protokoll bestimmt zu einem Anderen Kanal gehören (Zustand *„RELATED“*). Alle neu initiierten Verbindungen (Zustand *„NEW“*), werden in der 4. Regel nur dann zugelassen, wenn sie aus dem Intranet stammen, also **nicht** über das Interface *ppp0* hereinkommen. Die ausgehenden Verbindungen (*„OUTPUT“*) werden rigoros als vertrauenswürdig betrachtet und somit komplett freigegeben.

```
# Zuerst alle Regeln loeschen
/sbin/iptables -F INPUT

# Die Policy DROP wird eingeschaltet
/usr/sbin/iptables -P INPUT DROP

# Antworten werden zugelassen
/usr/sbin/iptables -A INPUT -m state \
 --state ESTABLISHED,RELATED -j ACCEPT

Neue Verbindungen nur vom Intranet
/usr/sbin/iptables -A INPUT -m state \
 --state NEW -i ! ppp0 -j ACCEPT

# OUTPUT ist frei
/sbin/iptables -P output ACCEPT
```

3. Das interne Netz darf nur bis auf den Router des Providers durchgreifen, nicht weiter. Ausnahme: Rechner *.12 darf überallhin. Ob das interne Netz dabei maskiert werden soll,ist im Gegensatz zu dem älteren *ipchains* an dieser Stelle irrelevant. An diesem Beispiel kann das Filtern auf bestimmte Ports bzw. eines Port-Intervalls beobachtet werden, das mit den Optionen *--sport* (source port) und *--dport* (destination port) eingeschaltet erden kann. Auch an dieser Stelle müssen die potentiellen Antwortkanäle freigeschaltet werden.

```
# Policy ist DENY
/usr/sbin/iptables -P FORWARD DENY

# PC Nummer *.12 darf ueberallhin
/usr/sbin/iptables -A FORWARD -p tcp \
 -s 192.168.8.12/32\ -d 0/0 --dport 1:1023 \
 -j ACCEPT

# Alle duerfen nur auf den Provider-Router
/usr/sbin/iptables -A FORWARD -p tcp \
 -s 192.168.8.0/24 -d 195.32.16.1/32 \
 --dport 1:1023 -j ACCEPT
#
# Antworten werden zugelassen
/usr/sbin/iptables -A FORWARD -m state \
 --state ESTABLISHED,RELATED -j ACCEPT
```

Auch bei diesem Beispiel ist die Reihenfolge wichtig, weil die Regeln sequenziell bearbeitet werden. Wären die beiden letzten Regeln vertauscht, könnte der Rechner mit der Knotennummer 12 auch nur auf den nächsten Router zugreifen, weil schon die allgemeinere Form auf seine Datenpakete zutrifft und die spezielle Regelung gar nicht mehr ausgeführt wird.

Da aber die Datenpakete, die den Forward-Filter passieren, auch noch maskiert werden müssen, steht noch die Konfiguration der POSTROUTING-Chain an. Im Gegensatz zu den ipchains wird hier also das komplette Filtersystem mit allen „echten“ Adressen konfiguriert, was übersichtlicher und einfacher ist. Das Masquerading stellt eine besondere Form des „**N**etwork **A**ddress **T**ranslation“ (NAT) dar, und stammt daher aus der Tabelle „NAT“. Um dem iptables-System dieses bekannt zu geben, muss die Table „nat“ über die Option *„-t nat“* **vor** der Benutzung des Targets eingeschaltet werden.

```
# Masquerading über ppp0
/usr/sbin/iptables -A POSTROUTING -t nat \
-o ppp0 -j MASQUERADE
```

Da das Masquerading lösgelöst von den eigentlichen Filterregeln konfiguriert wird, kann der Rest des Firewalls vor der „bösen Welt“ geschützt werden, ohne dass die Routingfunktionen in Mitleidenschaft gezogen werden. Somit ist eine Sicherheitslücke, die bislang bei den Firewall-

system der 2.2.x Kernels bestand, eliminiert worden.

Das obige Beispiel für den Zugriff aus dem Intranet ist natürlich nicht sonderlich sicher, da es nur auf Adressfilterung basiert. Mit Hilfe der Portangaben können aber zusätzlich bestimmte Dienste explizit zugelassen oder verboten werden. Soll z.B. der Rechner 192.168.8.12 für SSH-Verbindungen (Port 22, TCP) zum Firewall zugelassen werden, so kann das mit dieser Regel ermöglicht werden:

```
# Nur Rechner *.12 darf SSH ausfuehren
/usr/sbin/iptables -A INPUT -p tcp \
 -s 192.168.8.12/32 --dport 22 -j ACCEPT
#
/sbin/iptables -P output ACCEPT
```

Natürlich kann auf diese Weise jeder Dienst bereitgestellt werden, sofern der benutzte Port und das Protokoll bekannt sind.

Beachtenswert ist an dieser Stelle die freizügige Handhabe der OUTPUT-Chain, die via Policy für alles freigegeben wird. Im Gegensatz zu der ipchains immanenten Regelung, bleibt der Output-Filter lediglich für die Verbindungsaufbauten von den lokalen Prozessen verantwortlich. Weil jedoch die Prozesse des Firewalls als sicher eingestuft werden, kann diese Chain ohne extremes Risiko freigeschaltet werden. Handelt es sich jedoch um eine Arbeitsstation, an der ein Benutzer arbeitet, sollte auch die OUTPUT-Chain entsprechend begrenzt werden. Die Gefahr liegt hierbei im wesentlichen bei Trojanischen Pferden und Benutzern mit geringem Sicherheitsbewusstsein

Ein weiterer und häufig zitierter Sonderfall ist die FTP-Verbindung, weil sie nicht jeweils einen Port auf den verbundenen Rechnern benutzt, sondern je zwei. Daraus ergeben sich pro logischer FTP-Verbindung zwei Kanäle: Ein Steuer- (Port 21) und ein Datenkanal (Port 20). Der Client baut zunächst eine Verbindung zum Port 21 des Servers auf und teilt ihm eine Zieladresse mit. Anschließend erstellt der Server vom Port 20 aus eine Verbindung zum angegebenen Client-Port. Dieser Umstand trägt zu der Komplexität der benötigten Regelkombination bei. Die Schwierigkeit wird erhöht, wenn der FTP-Client ein maskierter Rechner ist, weil der Server nur die IP-Adresse des Firewalls „sieht“.

Um dieses Problem zu lösen, musste bislang ein spezielles Masquerading-Modul eingesetzt werden. Da in den iptables jedoch das Masquerading separat durchgeführt wird, ist das neu entstandene Modul „**Stateful Inspection**“ für diese Zwecke eingeführt worden. Dieses Modul entspricht den jüngsten Firewall-Konzepten und trägt maßgeblich zur Übersichtlichkeit und Flexibilität des Regelwerkes bei. In der folgenden Regel wird der Forwarding-Chain erlaubt, alle Datenpakete hindurchzulassen, die zu einer bereits aufgebauten TCP-Verindung gehören (*ESTABLISHED*), oder – wie im Fall des FTP – zu einem definierten Protokoll als zugehörig erkannt werden (*RELATED*). Auf diese Weise verringert sich die Anzahl der Firewall-Regeln drastisch, was die Lesbarkeit erheblich verbessert.

```
# Rechner *.10 darf FTP ausfuehren
/sbin/iptables -A FORWARD -p tcp \
-s 192.168.8.10/32 --dport 20,21 -j ACCEPT

/usr/sbin/iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Dieses Beispiel ist der Übersichtlichkeit halber unabhängig von bereits existierenden Regeln erstellt worden, sodass im Einzelfall geprüft werden muss, ob diese Regeln schon durch andere abgedeckt oder überlagert werden. Hier stellt sich wieder einmal das Problem der Reihenfolge, das nicht pauschal, sondern nur in Einzelfällen gelöst werden kann.

10.4 Automatisches Startup

Da alle Regeln nur im Hauptspeicher gehalten und bei jedem Bootvorgang neu gesetzt werden müssen, muss für diese Aufgabe eine separate Startup-Datei eingesetzt werden. Ein externes Skript besitzt außerdem den Vorteil, dass es bei SYS-V-konformen Distributionen direkt in die Startup-Verzeichnisse */etc/rc.d* (bzw. */sbin/init.d*) eingebunden werden kann.

Die beste Übersicht ergäbe eine Sortierung nach Zugehörigkeit zu einer logischen Regel, indem für eine Zugangsregel jeweils eine input- und eine output-Regel bzw zwei forward-Regeln direkt untereinander geschrieben würden. Aus den beschriebenen Gründen kann das leider nicht oder nur selten möglich sein, so dass die einig praktikable Lösung die Sprtierung nach Chains ist, die einen Kompromiss zwischen Lesbarkeit und technischen Anforderungen anstrebt. Die Konfigurationsdatei, die die entsprechenden Befehle enthält, kann so z.B. „*/etc/firewall.conf*“ genannt werden. Das nachfolgede Beispiel orientiert sich zwar an den Einstellungen, wie sie im obigen Text dargestellt werden, stellt aber eine andere Form der Reihenfolge der Regeln dar. Außerdem wurden allzu komplexe Strukturen der Übersichtlichkeit wegen weggelassen:

```
# Datei /etc/firewall.conf
# /sbin/sh

# Chain OUTPUT ########################
# Voreinstellungen fuer output
/usr/sbin/iptables -P OUTPUT ACCEPT
/usr/sbin/iptables -F OUTPUT

# Chain INPUT #########################
# Voreinstellungen fuer input
/usr/sbin/iptables -P INPUT DROP
/usr/sbin/iptables -F INPUT

# loopback-device freischalten
/usr/sbin/iptables -A INPUT -i lo -j ACCEPT

# Interne Kommunikation ermoeglichen
/usr/sbin/iptables -A INPUT \
 -s 192.168.8.0/24  -i eth0 -j ACCEPT

# Antworten werden zugelassen
/usr/sbin/iptables -A INPUT -m state \
 --state ESTABLISHED,RELATED -j ACCEPT

# Chain FORWARD #######################
# Forwarding Voreinstellungen
/usr/sbin/iptables -P FORWARD DROP
/usr/sbin/iptables -F FORWARD

# ppp0-Forwarding
# *.12 darf ueberall hin
/usr/sbin/iptables -A FORWARD \
 -s 192.168.8.12/32 -j ACCEPT

# Alle Anderen nur zum Proxy
/usr/sbin/iptables -A FORWARD -p tcp \
 -s 192.168.8.0/24 --sport 1024:\
 -d 195.32.16.1/32 -j ACCEPT

# Antworten werden zugelassen
/usr/sbin/iptables -A FORWARD -m state \
 --state ESTABLISHED,RELATED -j ACCEPT

# Chain POSTROUTING ###################
# Masquerading ueber ppp0
/usr/sbin/iptables -F POSTROUTING
/usr/sbin/iptables -A POSTROUTING -t nat \
-o ppp0 -j MASQUERADE
```

Da alle Einstellungen mit einem „Flush“ der jeweiligen „chain“ beginnen, kann diese Datei beliebig oft ausgeführt werden, ohne dass ungewollte Nebeneffekte entstehen.

Das Loopback-Device „lo“ muss auf jeden Fall freigeschaltet werden, weil sonst Dienste des Firewalls wie DNS, cron, at, lpd etc. nicht mehr arbeiten können.

Nun muss noch eine Datei erstellt werden, die für den automatische Start während des Boot-Vorganges sorgt.:

```
# /sbin/init.d/firewall
#
#!/bin/bash

case "$1" in
    start)
        echo "Starting firewalling."
        . /etc/firewall.conf
          echo "1" \
            >/proc/sys/net/ipv4/ip_forward
        ;;
    stop)
        echo -n "Shutting down firewalling:"
         /usr/sbin/iptables -P INPUT DROP
         /usr/sbin/iptables -F INPUT
         /usr/sbin/iptables -P OUTPUT DROP
         /usr/sbin/iptables -F OUTPUT
         /usr/sbin/iptables -F POSTROUTING
         echo "0" \
            >/proc/sys/net/ipv4/ip_forward
        ;;
    *)
        echo "Usage: $0 {start|stop}"
        exit 1
esac
```

Wenn das Skript mit dem Parameter „stop“ aufgerufen wird, werden **alle** Verbindungen gekappt, also auch die, die vom internen Netzwerk heraus aufgebaut werden. Der Rechner erlaubt daher nur noch lokale Logins.

10.5 DNAT und SNAT

Eine neu hinzugekommene Funktionalität der *iptables* ist das Network Adress Translation (NAT), das in einer speziellen Form bereits als Masquerading beschrieben wurde. Bei einem NAT werden die zu versteckenden IP-Adressen jeweils komplett durch eine andere ersetzt, jedoch nicht unbedingt durch die des Firewalls. Diese Funktion wird z. B. dann benötigt, wenn ein Server, der seine Dienste dem Internet anbieten soll, sich hinter der Firewall befindet, und somit nicht „sichtbar“ wäre. Die Firewall selbst leitet dann die entsprechenden Anfragen auf einen bestimmten Port einer bestimmten Adresse auf einen anderen Rechner, der sich durchaus im geschützten Bereich befinden kann.

Es wird dabei zwischen Source-NAT (SNAT) und Destination NAT (DNAT) unterschieden. Im ersteren Fall wird ein ankommendes IP-Paket auf einen anderen Port umgeleitet, während im letzteren Fall die echte IP-Adresse des Absenders ausgetauscht wird, um z.B. die wirklich Herkunft geheim zu halten. Das NAT kann nur außerhalb der Filter-Chains durchgeführt werden, also lediglich im PREROUTING (DNAT) oder im POSTROUTING (SNAT).

```
/usr/sbin/iptables -A PREROUTING -t nat \
 -p tcp -d 195.33.20.5 --dport 80 \
 -j DNAT --to-destination 192.168.10.1:8080
```

Die obige Regel fängt http-Anfragen auf den Knoten 195.33.20.5 ab und leitet sie automatisch

auf den Knoten mit der IP-Adresse 192.168.10.1 und den Port 8080. Auf diese Weise kann ein Webserver quasi über eine „Umleitung“ erreicht werden.

10.6 Transparent Proxy

Eine besondere Eigenschaft der NAT-Fähigkeiten des Kernels ist die Fähigkeit des „transparent proxy“, was prinzipiell nur eine Kombination aus Redirecting und Masquerading ist. Ein „normaler“ Proxy wie z.B. der Squid erwartet von den Clients, dass sie von der Existenz und der Benutzung des Proxies wissen, und entsprechend handeln. So muss z.B. der Netscape-Browser eines Clients im internen Netz die Information erhalten, dass er z.B. den Proxy auf dem Knoten 192.168.8.1 über den Port 8080 ansprechen muss, um eine Verbindung zu erhalten. Der Proxy erstellt seinerseits die „echte“ Verbindung zum gewünschten Server und gibt die zurückgesendeten Daten an den Client weiter. Der Vorteil dieser Lösung gegenüber eines reinen Paketfilters ist die Prüfbarkeit der Daten, da sie aktiv von dem Proxy weitergegeben werden müssen.

Bei einem „Transparent Proxy“ weiß der Client nichts über die Existenz eines Proxies. Nach dem obigen Beispiel sendet er wie gewohnt die Daten direkt an den anzusprechenden HTTP-Server über den Port 80 (HTTP). Im Kernel des Firewalls wird diese Anfrage auf den eigenen Port 8080 umgeleitet, hinter dem sich nun ein Proxy befindet. Dieser baut die Verbindung zu dem gewünschten Server auf und empfängt auch die gewünschten Daten. Diese Daten werden wiederum an den ursprünglichen Client weitergeleitet.

Der Client bemerkt nichts von der Zwischenstation des Proxies, was dazu führt, dass ein Client keine besonderen Fähigkeiten besitzen muss, um einen Proxy zu benutzen. Jedoch muss der auf dem Firewall eingesetzte Proxy mit den umgeleiteten Paketen umgehen können, was nicht selbstverständlich ist.

Prinzipiell wird diese Form der Weiterleitung mit dem Target „*REDIRECT*“ aktiviert, die ausschließlich für die input-chain zulässig ist. Um das obige Beispiel zu realisieren, kann dieser Befehl ausgeführt werden:

```
/usr/sbin/iptables -A PREROUTING -t nat \
 -p tcp -dport 80 -p tcp -i ppp0 \
 -j REDIRECT --to-port 8080
```

sofern nun ein Proxy auf den Port 8080 lauscht, wird ein Client, der den Port 80 eines beliebig anderen Rechners im Internet anspricht, automatisch auf 8080 des Firewalls umgeleitet.

Natürlich funktioniert dieser Mechanismus ebenfalls mit beliebigen Diensten eines bestimmten Rechners. Soll z.B. der Port 9090 des Firewalls auf den Port 23 (telnet) umgeleitet werden, so geschieht das durch diesen Befehl:

```
/usr/sbin/iptables -A PREROUTING -t nat \
 -p tcp --dport 9090 -i ppp0 \
 -j REDIRECT --to-port 23
```

Ist diese Regel eingetragen worden, kann von einem beliebigen Rechner der Telnet-Zugriff auf den Firewall über den Port 9090 erfolgen.

10.7 Protokollierung

Der beste Firewall ist so ziemlich nutzlos, wenn nicht in schriftlicher Form mitprotokolliert werden kann, ob und wie ein (verbotenes) Datenpaket in das lokale Netz hineingekommen ist oder es versucht hat.

Zu diesem Zweck gibt es die Protokollierungsfunktion, die ein passendes Paket im Syslog (/var/log/messages) festhält. Soll z.B. nach obigem Beispiel festgehalten werden, ob und wann der

Benutzer an Rechner 192.168.8.11 eine Telnet-Session aufbaut, muss der Befehl folgendermaßen aussehen:

```
/usr/sbin/iptables -A FORWARD -p tcp \
 -s 192.168.8.11/32 --dport 23 -o ppp0 \
 -j LOG
```

Das Target „*LOG*“ sorgt dafür, dass all das mitgeschrieben wird, das mit der Regel „matcht“. Im Gegensatz zu anderen Regeln ist die Auswertung nicht an dieser Stelle zu Ende, sondern wird bei der nächsten Regel fortgeführt. Diese Eigenschaft ist besonders dann wichtig, wenn mehrer sich gegenseitig ausschließende Regeln mitprotokolliert werden. Es können dabei beliebig viele Regeln in eine „chain“ geschrieben werden, ohne dass diese ein Auswirkung auf die wirklich aktive Regeln hätten.

Der folgende Befehl protokolliert z.B. alles, was von dem Rechner 192.168.8.11 zum oder über den Firewall hinaus gesendet wird:

```
/usr/sbin/iptables -A INPUT -p tcp \
 -s 192.168.8.11.2 --dport 22 -j LOG
```

Als Ergebnis würden in der syslog-Datei (z.B. /var/log/messages) Einträge wie der Folgende eingesetzt werden:

```
May 27 18:01:22 gandalf kernel: IN=eth0 OUT= MAC=00:00:e8:8c:46:bc:00:00:e8:8c:3
d:f5:08:00 SRC=192.168.8.11 DST=192.168.8.1 LEN=48 TOS=0x00 PREC=0x00 TTL=128 I
D=63263 DF PROTO=TCP SPT=1188 DPT=22 WINDOW=8192 RES=0x00 SYN URGP=0
```

Daraus lässt sich erkennen, dass von dem Rechner 192.168.8.11 von Port 1188 aus ein Paket gesendet wurde. Dieses Paket wurde über das Interface „eth0“ empfangen und hatte Port 22 (SSH) des Firewalls als Ziel. Bei diesem Paket handelt es sich um einen Verbindungsaufbau einer SSH-Session, die von dem Rechner 192.168.8.11 zum Firewall initiiert wurde.

Aber Vorsicht! wenn die Regel zu allgemein gehalten wird, führt das sehr schnell zu einem „Zumüllen“ der Log-Datei, was dazu führt, dass die wirklich relevanten Pakete übersehen werden. Es sollten also wirklich nur die Regeln mitprotokolliert werden, die als Ausnahme oder als Gefährdung gelten. Selbst dann können nur mittels besonderer Werkzeuge (grep, awk etc.) sinnvolle Informationen aus der Protokolldatei gewonnen werden. Eine Form der Vermeidung des „Datenmülls“ ist die Limitierung der protokollierten Pakete, die über das Zusatzmodul „limit“ erzwungen werden kann. Die obige Regel auf je 2 Pakete pro erstellter Verbindung müsste folgendermaßen lauten:

```
/usr/sbin/iptables -A INPUT -p tcp \
 -s 192.168.8.11.2 --dport 22 \
-m limit --limit-burst 2 -j LOG
```

10.8 Hilfswerkzeuge

Diese Art der Konfiguration ist natürlich für komplexe Lösungen recht unübersichtlich und fehleranfällig. Deshalb bieten mehrere Distributionen bereits übersichtlichere Lösungen über modularisierte Konfigurationsdateien an. Eine andere Lösung sind Administrationstools, die entsprechende Startup-Dateien automatisch generieren können.

Ein Tool, das ein wenig die Übersicht über die verfügbaren Optionen erleichtert ist der ***„knetfilter“***, ein grafisches Front-End für *„iptables“*, mit dem sich die notwendigen Regeln „zusammenklicken“ lassen. Es findest sich unter der URL:

http://expansa.sns.it/knetfilter

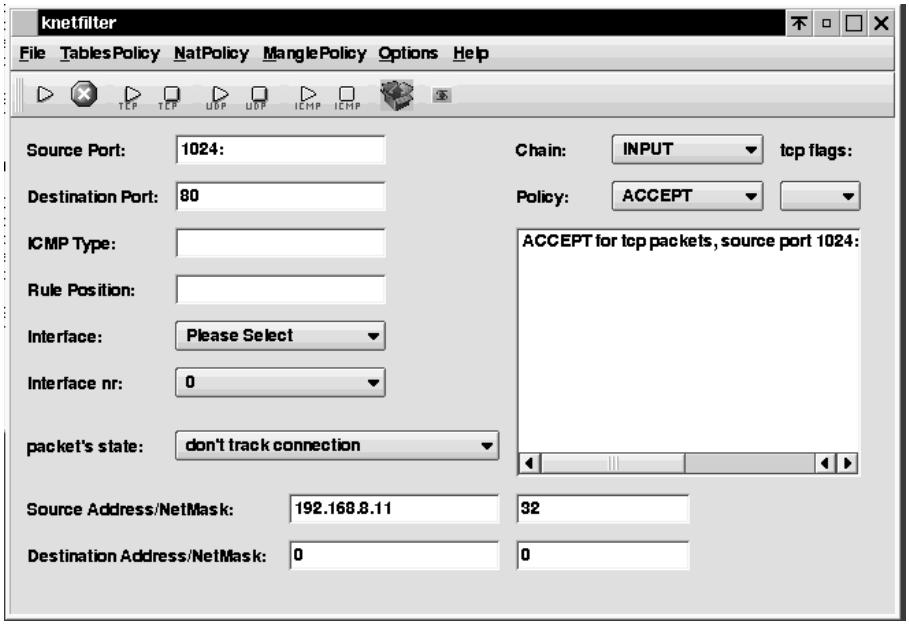


Abbildung 13: Konfiguration mit knetfilter

Das Tool selbst besteht aus einem Hauptdialog, in dem alle verfügbaren Optionen über Listen und Menüs auswählbar sind. Mit dieser Hilfe werden Syntaxfehler weitestgehend vermieden, doch leider werden weder eine übersichtliche Liste der vorhandenen Regeln noch eine logische Prüfung vollzogen, so dass die Hauptarbeit – die Konzeptionierung – nach wie vor manuell verrichtet werden muss.

Ein weiteres grafisches Tool ist der ***„Firewall Builder“***, der nicht nur die verfügbaren Optionen grafisch anbietet, sondern zusätzlich die eingesetzten Regeln in grafischer Form übersichtlich anzeigt. Einzelne Regeln können verändert und auch wieder gelöscht werden, daher bietet dieses Tool im Vergleich zu *knetfilter* eine gute Bearbeitbarkeit der vorhandenen Firewall-Konfigurationen. Der Firewall Builder ist unter

http://sourceforge.net/projects/fwbuilder/

zu erhalten. Weiterhin wird in Form eines Objekt-Baumes die Struktur der einzelnen Firewall-Objekte (Services, Chains, Tables etc.) ähnlich wie ein Classbrowser visualisiert. Auch diese Form erleichtert die Arbeit mit den Firewall-Regeln ungemein.

Trotzdem bleibt die Konzeptarbeit nach wie vor beim Administrator, der in herkömmlicher Form diese Regeln zusammenstellen muss. Der Firewall Builder stellt jedoch eine sehr gute Hilfe dar, werden doch die allzu häufigen Sytaxfehler durch diese Form der Generierung vermieden.

Um eine Schnell-Installation eines Firewalls durchzuführen, eignet sich der ***„FireStarter“***, der ebenfalls das Gnome-Toolkit benötigt, um gestartet werden zu können. Es unterstützt neben den „iptables“ auch die älteren „ipchains“, so dass es auch auf älteren Systemen eingesetzt werden kann.

Die Erfreuliche Eigenschaft dieses Tools ist der „Firewall-Wizard“, mit dem Standard-Lösungen sehr schnell erzeugt werden können. Leider können hier nicht detaillierte Regeln, sondern nur globale Zugänge/Verbote formuliert werden, was das Tool im Einzelfall unbrauchbar macht. Für einen ersten Entwurf ist es aber durchaus verwendbar

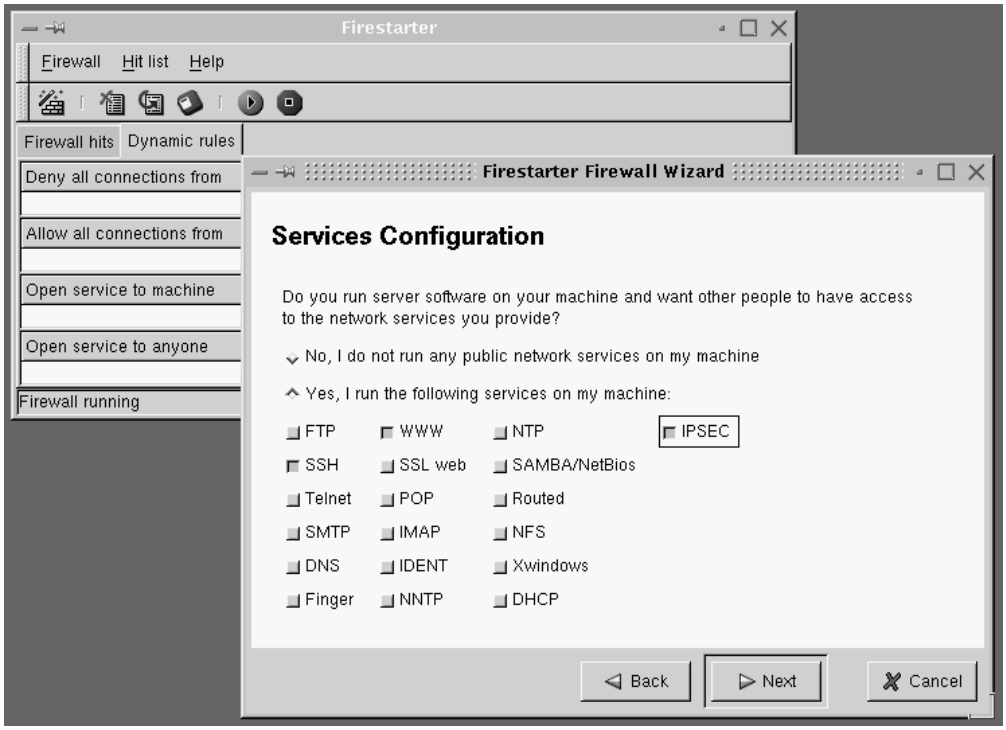


Abbildung 14: Konfiguration mit fwbuilder