7

4,40
Deutschland

KnowWare
Management!

Linux und Windows im Netzwerk

# Linux im Windows-Netzwerk

leicht & verständlich

KnowWare Management!

Kathrin Reeger & Eric Amberg

www.KnowWare.de

Deutschland: 4,40 EUR Österreich: 5,00 EUR
Schweiz: 8,60 SFR Luxemburg: 5,20 EUR

4 196468 404408 05007

## Bildqualität

**Wir versuchen die Dateigröße zu reduzieren, um die Downloadzeit zu verkürzen. Daher ist die Bildqualität in dieser Download-Datei nicht in allen Fällen optimal. Im Druck tritt dieses Problem nicht auf.**

## Acrobat Reader: Wie komme ich klar?

**F5/F6** öffnet/schließt die Ansicht **Lesezeichen**
**Strg+F** sucht

**Im Menü Ansicht stellst du ein, wie die Datei angezeigt wird**
**STRG+0** = Ganze Seite **STRG+1** = Originalgröße **STRG+2** = Fensterbreite

Im selben Menü kannst du folgendes einstellen: **Einzelne Seite**, **Fortlaufend** oder **Fortlaufend - Doppelseiten** ... Probiere es aus, um die Unterschiede zu sehen.

**Navigation**
**Pfeil Links/Rechts**: eine Seite vor/zurück
**Alt+ Pfeil Links/Rechts**: Wie im Browser: Vorwärts/Zurück
**Strg**+**+** vergrößert und **Strg**+**–** verkleinert

## Bestellung und Vertrieb für den Buchhandel

KnowWare-Vertrieb, Postfach 3920, D-49029 Osnabrück
Tel.: +49 (0)541 33145-20 Fax: +49 (0)541 33145-33
bestellung@knowware.de
www.knowware.de

## Autoren gesucht

Der KnowWare-Verlag sucht ständig neue Autoren. Hast du ein Thema, dass dir unter den Fingern brennt? - ein Thema, das du anderen Leuten leicht verständlich erklären kannst?

Schicke uns einfach ein paar Beispielseiten und ein vorläufiges Inhaltsverzeichnis an folgende Adresse:

lektorat@knowware.de

Wir werden uns deinen Vorschlag ansehen und dir so schnell wie möglich eine Antwort senden.

**www.knowware.de**

# Inhaltsverzeichnis


**Linux und Windows zusammen – passt denn das? ..... 5**
Was erwartet dich in diesem Heft? ..... 5
Was solltest du mitbringen? ..... 5
**Bevor wir loslegen … ..... 6**
Das Szenario ..... 6
Bytes aus dem Reagenzglas ..... 6
Testen von außerhalb ..... 7
… welches Linux nutzen wir? ..... 7
**Linux „scharf schalten“ ..... 7**
Laborkonfiguration ..... 7
Grundeinstellungen ..... 7
Konfiguration testen ..... 8
Was steckt dahinter? ..... 8
Jeder Rechner hat einen Namen ..... 10
Was steckt dahinter? ..... 10
Router – Die Pfadfinder ..... 11
… und noch mehr Background ..... 12
**Samba – der Linux-Datei- und Druckserver für Windows ..... 13**
Samba installieren ..... 13
Den Samba-Daemon starten ..... 13
Was steckt dahinter? ..... 14
Automatischer Start von Samba ..... 14
Den Dienst testen ..... 14
Den Samba-Dienst testen ..... 15
**Samba Grundkonfiguration ..... 16**
Die Sektion [global] ..... 16
Die erste Freigabe ..... 16
Den Windows-Rechner anpassen ..... 17
Test der Freigabe ..... 17
Mit Freigaben arbeiten ..... 18
Alternative Zugriffsmöglichkeiten ..... 18
Was steckt dahinter? ..... 19
**Die „echte“ Samba-Umgebung ..... 21**
Gruppen einrichten ..... 21
Benutzer einrichten ..... 21
Rechte festlegen ..... 21
Weitere Verzeichnisse anlegen ..... 21
Security = user ..... 22
smbpasswd ..... 22
Die Freigaben erstellen ..... 22
Die Freigaben testen ..... 22
Die Home-Verzeichnisse ..... 23
Angepasste Benutzerkennung ..... 24
Benutzer unter Windows anlegen ..... 24
CD-ROM-Freigabe ..... 24
Optional: guest ok = yes ..... 24
Netzwerkdrucker freigeben ..... 25
Windows-Netzwerkdrucker ..... 25

**SWAT – Samba über den Browser konfigurieren ..... 26**
SWAT starten ..... 26
Ein SWAT-Workshop ..... 27
SWAT nutzen von Windows-1 aus ..... 29
**Zusammenfassung: Aufbau unseres Familienservers ..... 30**
Konzept erstellen ..... 30
Benutzer und Gruppen erstellen ..... 30
Verzeichnisse erstellen ..... 30
Eigentümer und Rechte setzen ..... 30
Drucker einrichten ..... 30
Windows-Clients konfigurieren ..... 30
Samba konfigurieren ..... 30
**Samba als Domänen-Controller ..... 31**
Computer-Konten einrichten ..... 31
Einen Domänen-Admin erstellen ..... 31
Logon- und Profil-Verzeichnisse ..... 31
Samba als PDC konfigurieren ..... 31
Windows in die Domäne bringen ..... 32
Das Anmeldeskript ..... 33
Servergespeicherte Profile ..... 33
Backup ..... 33
**SSH – die sichere Fernwartung auf der Konsole ..... 34**
Telnet versus SSH ..... 34
Das Konzept von SSH ..... 34
Unter Linux mit SSH verbinden ..... 35
Unter Windows mit SSH verbinden ..... 35
**Linux als DSL-Router ..... 37**
Zur Lage der Nation ..... 37
… und ISDN? ..... 37
Hardware ..... 37
Installation ..... 37
Netzwerkkarten konfigurieren ..... 37
Einen DNS-Server einrichten ..... 39
Alternativen ..... 41
**Apache – der Web-Indianer ..... 41**
Wo kommt der Webserver hin? ..... 41
Apache installieren ..... 41
Apache testen ..... 42
Apache konfigurieren ..... 42
Was steckt dahinter? ..... 44
**FTP – Dateien über das Internet übertragen ... 45**
FTP installieren ..... 45
vsftp konfigurieren ..... 45
FTP testen ..... 45
Was steckt dahinter? ..... 46
**Aufbau einer DMZ ..... 47**
Wozu eine DMZ? ..... 47
Eine DMZ mit nur einer Firewall ..... 47
Lab 3 – DMZ ..... 48
**iptables als Netzwerkfirewall ..... 49**
Was ist eine Netzwerkfirewall? ..... 49
Die FORWARD-Kette ..... 49
Das DSL-Router-Firewall-Skript ..... 49
Wie funktioniert das Skript? ..... 50
**DynDNS.org – immer erreichbar über den eigenen Namen ..... 55**
Anmeldung bei DynDNS.org ..... 55
Dynamische IP-Aktualisierung ..... 56
**Anhang: Netzwerksniffer ..... 57**
Ethereal ..... 57
tcpdump – Sniffer auf der Konsole ..... 59
**Schlusswort ..... 59**
**Stichwortverzeichnis ..... 61**

# Linux und Windows zusammen – passt denn das?

… wenn es nach Microsoft ginge, wohl eher nicht. Aber zum Glück tanzt nicht alles nach der Pfeife von Old Bill (Gates). Zwar ist Linux für Microsoft ein Dorn im Auge – umgekehrt gilt das aber keineswegs. Jeder realistische Linux-Freund weiß, dass Windows im Desktop-Bereich auf absehbare Zeit nicht zu verdrängen ist – weder von Linux noch von sonst wem …

Warum also nicht die Vorteile beider Systeme vereinen? Linux hat sehr viele Schnittstellen zu Windows. Am besten eignet sich Linux als Server-Betriebssystem in einem Windows-Netzwerk. Da du dieses Heft in den Händen hältst, möchtest du offensichtlich mehr darüber erfahren; herzlich willkommen also in der Welt heterogener Netzwerke!

## Was erwartet dich in diesem Heft?

Wir haben in diesem Heft viele schnucklige Sachen vor:

1. Zunächst bauen wir einen *Samba-Server* auf – einen Datei- und Druckserver für Windows-Netzwerke.
2. Auch als *zentraler Anmeldeserver* (PDC), wie wir sehen werden, macht Samba eine gute Figur.
3. Desweiteren erfährst du, wie du Samba über den Browser konfigurierst – das Stichwort heißt *SWAT*!
4. Dann bauen wir einen *Linux-DSL-Router* auf – so können endlich alle ins Internet, ohne dass ein bestimmter Arbeitsplatzrechner laufen muss. Nutzt du ISDN, kommst du hier ebenfalls auf deine Kosten!
5. Schließlich werden wir einige typische Dienste unter Linux aufsetzen, die du auch mit Windows-Clients nutzen kannst:
   - *SSH* zur sicheren Fernverwaltung über die Konsole.
   - *Apache* als Webserver für deinen Intra- und Internetauftritt.
   - *vsftp* als FTP-Server zur Datenübertragung

   Hiermit kannst du schon ziemlich professionell mit Linux im Netzwerk arbeiten.
6. Mit unserer Internet-Präsenz wollen wir dann natürlich auch sicher ins Internet. Daher bauen wir uns eine *DMZ* (entmilitarisierte Zone) für unseren Linux-Server auf.
7. Fehlt noch etwas? Ja: eine gute *Firewall-Konfiguration*, damit wir die Kommunikation von und zu deinem Server und deinem internen Netzwerk auf das nötige beschränken – wir wollen schließlich keinen unerwünschten Besuch haben, oder?
8. … und schließlich wollen wir auch bei einer Dial-In-Verbindung immer über dieselbe Adresse erreichbar sein; also richten wir uns *DynDNS* ein.

Das ist ein ziemlich sportliches Programm – nix für „Mausschubser“ ☺. Hast du aber Lust, dich mit Linux und seinen Netzwerkfähigkeiten näher zu befassen und auch ein wenig Hintergrundwissen und Netzwerk-Know-how „abzugreifen“, bist du hier genau richtig.

## Was solltest du mitbringen?

Wir wollen die Leser der Vorgänger-Hefte nicht langweilen – daher bauen wir auf einigen Linux-Grundlagen auf, die du aber im jeweiligen Heft unter der angegebenen Seitenzahl nachschlagen kannst.

Ach ja! Noch eine wichtige Voraussetzung:

Du brauchst ein wenig Geduld!

*Linux im Netzwerk* ist kein Thema für einen Nachmittag – es gibt jede Menge Stolperstellen, und auch du wirst nicht allen Fallen auf Anhieb ausweichen können. Das kann dir selbst der dickste Wälzer nicht ersparen – von einem 60-Seiten-Heft ganz zu schweigen …

Wir haben uns aber alle Mühe gegeben, den Inhalt so verständlich und unterhaltsam wie möglich zu präsentieren, damit du maximalen Gewinn aus dieser Lektüre ziehst. Also: Fröhliches „Netzwerkeln“ mit Linux …

… wünschen

Eric und Kathrin

# Bevor wir loslegen …

… werfen wir einen Blick auf die Rahmenbedingungen:

## Das Szenario

Um die Sache anschaulich zu machen, verpacken wir sie in eine lebendige Geschichte. Gehen wir davon aus, dass du weiblichen Geschlechts bist. Dann stell dir folgendes vor (sonst entsprechend andersherum):

Du hast einen Mann und zwei Kinder: Anna (14 Jahre) und Paul (12 Jahre). Natürlich hat jeder seinen eigenen PC – anders wäre es undenkbar (Anna kann es gar nicht leiden, wenn Paul wieder irgendwas an ihrem Computer verstellt und dann behauptet, er hätte gar nix gemacht!).

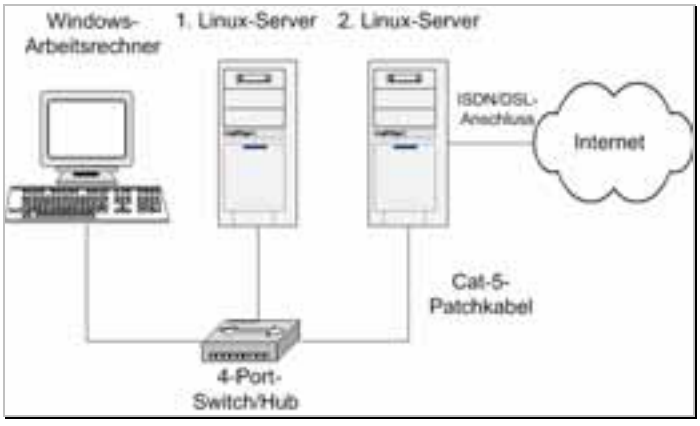
Genau wie du benötigt auch dein Mann ein Notebook für seine Dienstreisen, das ab und zu an das Netzwerk angeschlossen werden muss. Insgesamt habt ihr also 6 Computer in eurem Heim-Netzwerk.

Bisher ist nur dein Rechner über ISDN ans Internet angeschlossen. Die anderen Familienmitglieder können dank *Jana-Proxy-Server* (www.janaserver.de ) über deinen Computer surfen – allerdings nur, wenn dieser auch eingeschaltet ist. So richtig schnell ist das dann auch nicht – 64 KBit/s mit ISDN ist eben nicht die Welt.

Letzte Woche musstest du Annas Rechner neu installieren, weil die Festplatte einen Schuss weg hatte – alle Daten waren verloren, weil Anna keine Sicherung gemacht hatte. Sie hat sich natürlich schwarz geärgert und gefragt, ob Mama nicht irgendwas machen könnte, damit die Daten automatisch gesichert werden würden …

Soweit die Ausgangssituation. Wie du siehst, gibt es einigen Handlungsbedarf – und es kommt im Laufe unserer Geschichte noch einiges dazu, warte es ab!

Du möchtest ein Netzwerk in deiner Firma aufbauen und hättest lieber ein Szenario aus diesem Bereich? Keine Sorge – alles, was wir besprechen, kannst du fast *eins-zu-eins* auf eine kleine Firma übertragen – nur sind es vielleicht etwas anders gelagerte Interessen, die zu der einen oder anderen Änderung im Netzwerk führen.




Laut und deutlich vernehmen wir deinen nächsten (gedanklichen) Einwand: Du hast keine 6-8 Rechner zu Hause? Na dann lies weiter …

## Bytes aus dem Reagenzglas

Soll eine Konfiguration getestet werden (man spricht auch von *Proof-of-Concept*, zu deutsch: Konzeptüberprüfung), wird auch in der Praxis zunächst eine Laborumgebung (engl. *Lab* genannt) aufgebaut, eine Art Modell der Realität. Hier kann man dann nach Herzenslust experimentieren; und es macht gar nichts, wenn du aus Versehen auf den falschen Knopf drückst und alles zerschießt – es wird einfach wieder aufgebaut.

Um das hier vorgestellte Szenario im Lab nachzubauen, benötigst du insgesamt drei Rechner:

- einen ersten Linux-Server, an dem wir uns u.a. mit SAMBA austoben werden,
- einen zweiten Linux-Server, der unser Tor ins Internet wird, und
- einen Windows-Client – am besten *Windows XP Professional*, es geht aber auch mit anderen Windows-Versionen. Dieser PC steht für alle Client-Rechner im Netzwerk.

Drei Rechner hast du auch nicht? Macht nichts! So lustig es klingt, die meisten Serverdienste kannst du auf einem einzelnen Rechner aufsetzen und austesten!

Sinnvoller wäre es aber, wenn du wenigstens *zwei* Rechner für unser Lab erübrigen kannst – z.B. deinen normalen Arbeitsrechner als Client und einen Zweit-PC.

> Der Zweit-PC sollte zum mindesten ein P II 300 mit 128 MB RAM und einer 3 GB Festplatte sein. Das wird dann unser Linux-Server und damit (vorerst) unser Spielzeug.
>
> Für einen *Linux-Router* (später mehr dazu) sind die Anforderungen noch geringer: P I 200 Mhz, ein paar hundert MB Festplattenplatz und 64 KB RAM. Eigentlich geht noch weniger, aber dann müssten wir tricksen.

Vielleicht hast du so einen alten Rechner noch irgendwo in der Ecke zu stehen? Dann werden wir ihn später als DSL-Router einer sinnvollen Bestimmung zuführen.

Außerdem benötigst du für jeden Rechner eine Netzwerkkarte – für unseren Router sogar zwei bzw. drei, wenn du eine DMZ (siehe Seite 47) aufbauen möchtest. Aber keine Bange: einfache 10/100 MBit-Netzwerkkarten kosten nicht viel mehr als 10 Euro.

Verbunden werden unsere Rechner über *Cat 5-Patch-Kabel* und einen *4-Port-Switch* oder *Hub*. Damit sieht unser komplettes Lab erst einmal so aus wie in der Abbildung.

Je nach konkreter Zielsetzung passen wir die Konfiguration an. Stehen nur zwei Rechner zur Verfügung, übernimmt der zweite abwechselnd die eine oder die andere Funktion. Wir werden das dann im Einzelnen sehen.

### Testen von außerhalb

Im Verlauf des Heftes richten wir einige Dienste ein, die vom Internet aus erreichbar sein sollen. Das musst du mit einem Zugriff von außerhalb prüfen können.

Eine Möglichkeit: Bitte einen Freund, den Dienst zu testen. Er kann dir auch telefonisch „live“ Bericht erstatten. Klappt etwas nicht, könnt ihr gemeinsam „debuggen“, das heißt, den Fehler suchen.

*Tipp*: Wenn du ISDN in Verbindung mit DSL hast, bist du vielleicht stolzer Besitzer einer ISDN-Karte?! Unter diesen Umständen kannst du auch über die ISDN-Karte mit einem (anderen) Rechner ins Internet gehen und auf den DSL-Router (der über DSL angeschlossen ist) zugreifen – so findet tatsächlich ein Internetzugriff statt, auch wenn du dich genau neben deinem DSL-Router befindest …

So, nun aber genug der Vorrede – machen wir Linux netzwerkfähig! Hoppla …

### … welches Linux nutzen wir?

… ach ja: Wir nutzen *SuSE 9.1*. Du kannst im Prinzip auch jede andere Distribution nutzen, z. B. *Mandrake* oder *Debian*. Allerdings werden wir uns des (u.E. sehr guten) Administrations-Tools YaST bedienen, mit dem wir die Netzwerkkarten, ISDN und DSL konfigurieren.

Jede Distribution bringt ihre eigenen Tools mit, die wir leider nicht alle besprechen können. Aber die meisten besprochenen Dinge sind distributionsübergreifend!

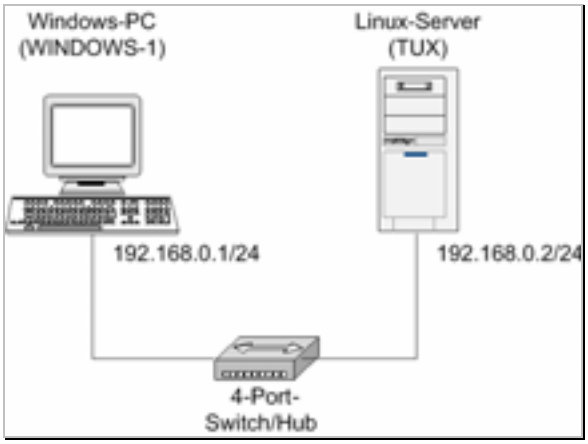
## Linux „scharf schalten“

Zurück zu unserem Szenario: Du hast dich entschlossen, einen zentralen Dateiserver aufzubauen, damit du alle wichtigen Daten zentral sichern kannst. Also hast du dir einen PC organisiert, auf dem du Linux installiert hast. Jetzt brauchen wir erst mal eine Netzwerkkonfiguration, damit Linux überhaupt im Netz „mitspielen“ kann.

Hast du die beiden Vorgänger-Hefte gelesen, betrachte diesen Abschnitt als kleine Auffrischung.

### Laborkonfiguration

Möchtest du das Folgende im „Lab“ nachstellen, sollte deine Konfiguration in etwa so aussehen:




**Unser erstes Lab**

Gib dem Windows-Rechner die IP-Adresse 192.198.0.1, Subnetzmaske 255.255.255.0. Die Darstellung `<IP-Adresse>/24` besagt genau das. Dazu später mehr.

Außerdem soll dein Windows-Rechner `WINDOWS-1` heißen. Wie du die Einstellungen unter Windows vornimmst, weißt du sicherlich.

Stelle sicher, dass keine Personal Firewall wie z. B. *Zone Alarm* aktiviert ist.

Nun widmen wir uns dem Linux-Server.

### Grundeinstellungen

Nutzt du SuSE, kannst du die Konfiguration der Netzwerkkarte über YaST2|NETZWERKGERÄTE|NETZWERKKARTE ändern.

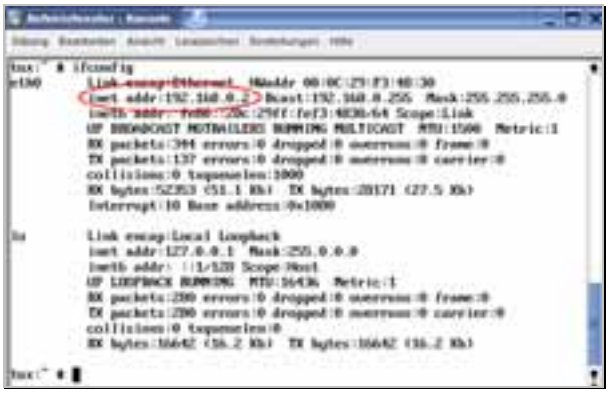

Unter ÄNDERN|BEARBEITEN wählst du KONFIGURATION DER STATISCHEN ADRESSE und trägst hier eine passende IP-Adresse ein – für unser Szenario die 192.168.0.2. Vergiss nicht die Subnetzmaske (255.255.255.0) einzutragen!


Dies ist die minimale Konfiguration, um im Netzwerk kommunizieren zu können. Über WEITER|BEENDEN wird die Konfiguration aktiviert.

Die Distributionen unterscheiden sich hier wie schon erwähnt voneinander. Aber jede der großen Linux-Distributionen bringt ein Administrationstool zur Netzwerkkonfiguration mit.

## Konfiguration testen

Kann dein Linux-Server jetzt tatsächlich schon kommunizieren? Wir werden es sehen! Wechsle auf eine Konsole. Als Erstes prüfen wir, ob die Konfigurationseinstellungen wirklich übernommen wurden. Dazu existiert unter Linux der Befehl `ifconfig`:



**Mit ifconfig die Netzwerkkonfiguration überprüfen**

In der Spalte *links* stehen die Netzwerk-Schnittstellen. Unter `eth0` (das ist die erste Ethernet-Karte) siehst du folgendes:

```
inet addr: 192.168.0.2
```

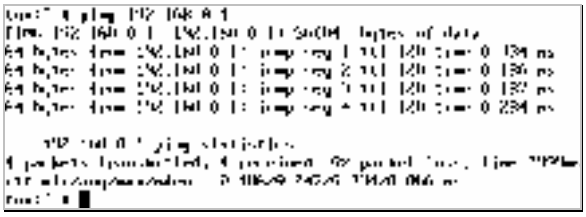
Scheint also alles fein zu sein.

Nächster Test: Wir „pingen“ einen anderen Rechner an – im Lab ist das unser Windows-PC:

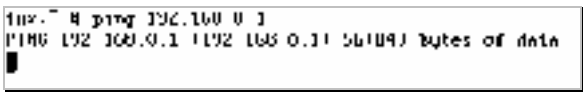
```
ping 192.168.0.1
```

Bei Windows werden standardmäßig vier „Pings“ gesendet; unter Linux musst du den Ping mit Strg + C beenden, soll er nicht ewig laufen.

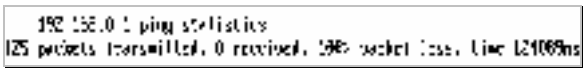
Antwortet der andere Rechner, sieht das so aus:



Funktioniert es nicht, zeigt Linux nichts an:



Erst mit Strg + C erhältst du eine Statistik:



`100% packet loss` bedeutet, dass keine Ping-Anfrage beantwortet wurde. Nicht gut, das!

Kontrolliere in diesem Fall Folgendes:

- IP-Adressen richtig eingegeben?
- Subnetzmaske korrekt?
- Ping auf die eigene IP-Adresse möglich?
- Personal Firewall deaktiviert?

Die Kommunikation über Ping muss zunächst funktionieren, bevor wir weiter machen!

## Was steckt dahinter?

Wie versprochen wollen wir uns etwas Hintergrundwissen aneignen, damit du weißt, an welchen „Schrauben“ du drehen musst, wenn etwas nicht funktioniert.

### TCP/IP

Jede Kommunikation, an der Linux beteiligt ist, basiert auf TCP/IP, dem Internet-Protokoll-Stapel. Es handelt sich um eine Sammlung von z.T. aufeinander aufbauenden Protokollen, die jeweils eine eigene Funktion haben.

Die Basisprotokolle sind IP (Internet Protocol) und TCP (Transmission Control Protocol). IP lernst du im nächsten Abschnitt kennen; TCP stellen wir später vor.

### IP – das Internetprotokoll

IP, das unterste Basisprotokoll, ist eigentlich immer mit von der Partie, wenn es um Netzwerkkommunikation geht. Über IP werden die Rechner adressiert. Außerdem wird praktisch jedes Datenpaket mittels IP transportiert – dies sind sozusagen die Räder eines gedachten Fahrzeugs zum Transport der Nutzlast bzw. der Nutzdaten.

### Die IP-Adresse

IP-Adressen bestehen aus vier mit Punkten getrennten Bytes. Jedes Byte kann einen Wert zwischen 0 und 255 annehmen; im ersten darf aber nicht 0 stehen.

Gültige IP-Adressen sind demnach:

- 192.168.0.1
- 172.16.5.13
- 212.255.255.47

Keine gültigen Adressen sind:

- 0.13.251.17
- 192.275.360.24

IP-Adressen sind in logische Netzwerke gruppiert. Jede IP-Adresse ist in einen Netz- und einen Hostanteil unterteilt. Der Netzanteil steht immer links.

| Netzanteil | | | Hostanteil |
|---|---|---|---|
| 192 | 168 | 0 | 1 |

Als *Host* wird jeder am Netzwerk angeschlossene Rechner bezeichnet, egal welche Funktion er hat.

Alle IP-Adressen in einem logischen Netz haben *dieselben* Werte im Netzanteil; der Hostanteil dagegen muss (im jeweiligen Netz) für jeden Rechner *eindeutig* sein.

## Die Subnetzmaske

Damit der Computer weiß, wo die Trennung zwischen Netz- und Hostanteil vorgenommen wird, gibt es die Subnetzmaske. Sie besteht ebenfalls aus vier Byte, wird aber nach anderen Regeln gebildet. Sie wird auf die IP-Adresse gelegt und „verdeckt" einen Teil der Adresse.

Vereinfacht gesagt: Überall da, wo 255 steht, ist Netz-, wo 0 steht, ist Hostanteil.

| Netzanteil | | | Hostanteil |
|---|---|---|---|
| 192 | 168 | 0 | 1 |
| 255 | 255 | 255 | 0 |

Möchtest du mehr erfahren, lies weiter; andernfalls kannst du den eingerahmten Text überspringen.

**Profi-Know-how**: Eigentlich arbeitet ein Computer *binär*, d.h. mit Bits. Ein Bit kann den Zustand 0 (nicht gesetzt) oder 1 (gesetzt) annehmen. Ein Byte besteht aus 8 Bits, die jeweils einen bestimmten Wert haben:

```
Bit:     8   7   6   5   4   3   2   1
Wert:  128  64  32  16   8   4   2   1
```

Damit würde 173 *dezimal* folgendem (binären) Bitmuster entsprechen:

```
10101101
```

*Erklärung:* Die entsprechenden Bits werden in ihrer Wertigkeit addiert:

```
Bit:    1   0   1   0   1   1   0   1
Wert: 128      32       8   4       1
```

Alle *gesetzten* Bits werden berücksichtigt:

128 + 32 + 8 + 4 + 1 = 173.

Voilà! Sind alle Bits in einem Byte gesetzt, ergibt dies 255. Damit kann ein Byte 256 verschiedene Zustände annehmen: 255 + 0 (kein Bit gesetzt).

Für die Subnetzmaske gilt, dass von links durchgängig alle Bits bis zur Grenze zwischen Netz- und Hostanteil der IP-Adresse gesetzt sind. Das sieht für 255.255.255.0 dann so aus:

```
11111111.11111111.11111111.00000000
```

Es sind also 3 x 8 = 24 Bits gesetzt. Daher schreibt man oft für *IP: 192.168.0.1 Subnetz 255.255.255.0* einfach *192.168.0.1/24*.

## Subnetz, Hosts und Broadcast

Die IP-Adresse bezeichnet die Adresse eines Rechners. Sie ist eindeutig, darf also insgesamt nur einmal vorkommen. Das *Subnetz* selbst, also das, was durch den Netzanteil definiert ist, entspricht der allerersten möglichen Adresse in einem Netz. Das ist meistens die Null. UUnser Subnetz lautet also: 192.168.0.0/24.

Die allerletzte Adresse in einem Netz dagegen bezeichnet man als *Broadcast* (zu deutsch: Rundruf). Damit „brüllt" ein Rechner in das eigene Subnetz und spricht nicht nur einen bestimmten, sondern alle Hosts an. Das ist so, als würdest du in einen Raum mit vielen Menschen gehen und laut nach Herrn Müller fragen – alle hören deine Frage, aber nur Herr Müller wird mit einem „Ja?" antworten. Broadcasts bleiben im eigenen Subnetz und werden nicht in andere Subnetze weitergeleitet.

Host-Adressen sind alles zwischen Netz- und Broadcast-Adresse, also 192.168.0.1 bis 192.168.0.254.

## Netzklassen

IP-Netze werden (historisch) in Klassen unterteilt:

| Netzklasse | erstes Byte | Subnetzmaske |
|---|---|---|
| A | 1-127 | 255.0.0.0 |
| B | 128-191 | 255.255.0.0 |
| C | 192-223 | 255.255.255.0 |
| D | 224-239 | n.a. |
| E | 240-255 | n.a. |

In der Praxis werden nur die ersten drei Klassen verwendet. Klasse D wird für spezielle Netzwerkanwendungen (*Multicast* genannt) genutzt, Klasse E ist experimentell.

Sprechen wir also von einem Klasse-C-Netzwerk, meinen wir ein Netzwerk, bei dem das erste Byte irgendeinen Wert zwischen 192 und 223 und (viel entscheidender) die Subnetzmaske den Wert 255.255.255.0 hat. Das Netzwerk, in dem wir unsere Test-Rechner erstellt haben, ist ein Klasse-C-Netzwerk (192.168.0.0/24).

Zum Vergleich sieht ein Klasse-B-Netzwerk so aus:

172.16.0.0 Subnetzmaske 255.255.0.0

oder kurz: 172.16.0.0/16.

Die Hostadressen reichen hier von 172.16.0.1 bis 172.16.255.254, weil ja die letzten *zwei* Bytes für die Host-Adressierung zur Verfügung stehen. Die Broadcast-Adresse in diesem Netz ist 172.16.255.255.

## Ping – sind wir im U-Boot?

Ganz so abwegig ist der Vergleich nicht, wie du gleich sehen wirst:

Hinter dem Ping steht ein Protokoll namens ICMP (Internet Control Message Protocol). Es wird genutzt, um Kontroll- und Fehlermeldungen zwischen Netzwerkkomponenten auszutauschen. ICMP kennt verschiedene Meldungstypen, z.B.:

- ICMP Typ 0: Echo Reply (Ping Antwort)
- ICMP Typ 3: Destination unreachable (Ziel nicht erreichbar)
- ICMP Typ 8: Echo Request (Ping Anfrage)

Bei einem *Ping* werden ICMP Typ 8-Datenpakete (Echo Request) an den Zielhost verschickt. Der Zielrechner antwortet mit ICMP Typ 0 (Echo Reply). Wie du siehst, ist der Vergleich mit dem Echo-Radar eines U-Bootes nicht ganz aus der Luft (bzw. dem Wasser) gegriffen...

Typ 3 wird übrigens für den Fall verwendet, dass im Internet ein Router (eine Verbindungskomponente zwischen Netzwerken) feststellt, dass das Ziel eines Datenpaketes nicht erreichbar ist.

### Ethereal – Für den Hacker in dir …

Kann man das mit dem Ping und so eigentlich auch mal sehen oder ist das nur theoretisches Geschwafel?

Na, aber …! Das kann man sogar detailliert nachvollziehen – und zwar mit einem Netzwerk-Sniffer. Damit lässt sich jedes Datenpaket von und zu einem Rechner aufzeichnen und bis aufs Byte genau analysieren.

Der wahrscheinlich beste Open-Source-Netzwerk-Sniffer ist *Ethereal* (www.ethereal.com). Er ist für Windows und Linux erhältlich.

Wenn du Interesse hast, kannst du jetzt zur Seite 57 springen, wo wir dir die Installation und Anwendung von *Ethereal* für Linux und Windows zeigen. Wir können dir den Einsatz dieses Tools wärmstens empfehlen – es eignet sich hervorragend zur Fehlersuche!

## Jeder Rechner hat einen Namen

… so auch dein Linux-Server. Und diesen bekommt er über die Netzwerkkonfiguration. Öffne wiederum YaST2 und wähle NETZWERKGERÄTE|NETZWERKKARTE. Unter ÄNDERN|BEARBEITEN findest du den Button RECHNERNAME UND NAMESERVER.


Gib hier als Rechnername *tux* und als Domainname *linux.local* ein.

*Tux* heißt übrigens der Linux-Pinguin.

Die Zusammenhänge erfährst du im nächsten Abschnitt.

## Was steckt dahinter?

Du hast soeben den Host- und DNS-Namen für dein System vergeben. Die Syntax ist:

`<Hostname>.<Domain>.<Topleveldomain>`

Bei uns ist das `tux.linux.local`.

Dieser „Namensraum“ wird auch im Internet verwendet. Da heißen die Rechner aber meistens `www` – aus Konventionsgründen. So ist also `www.knowware.de` auch ein vollständiger Domainname. Die eigentliche Domain bezeichnet meistens den Inhalt oder Namen einer Firma bzw. Website, wie z.B. `www.microsoft.com`, `www.linux.org` usw.

*Merkst du was?* Die Endung, also die so genannte Topleveldomain (TLD), ist unterschiedlich. Die meisten TLDs sind Länder-Codes (z.B. `.de`, `.nl`, `.fr` usw.); andere sind auf den Inhalt bezogen, wie z.B. `.org` (Organisation), `.com` (Commercial – kommerziell) oder `.gov` (Government – US-Regierung). Für unsere privaten Zwecke wählen wir `.local`. Das gibt es nicht im Internet.

### Der Windows-Name

Wie du sicher weißt, musst du auch einem Windows-Rechner einen Namen geben. Diesen findest du in den SYSTEMEIGENSCHAFTEN. Da kommst du z.B. per Rechtsklick auf den *Arbeitsplatz* hin. Das Dialogfeld versteckt sich hinter der Option EIGENSCHAFTEN:


Wie du siehst, heißt unser Computer hier `Windows-1`. Er ist der Arbeitsgruppe MSHEIMNETZ zugeordnet. Das kannst du über die unten stehenden Buttons anpassen – dazu mehr auf Seite 17.

### Vergleich: Linux- und Windowsname

Ist das bei Linux nun das Gleiche wie bei Windows? Die eindeutige Antwort lautet: Es kommt darauf an …!

Grundsätzlich nutzt Windows zunächst einmal nicht DNS sondern den NetBIOS-Namensraum. Hier gibt es nur einen einfachen Rechnernamen, der höchstens 15 Zeichen lang sein darf und bis auf das Minus (-) keine (offiziellen) Sonderzeichen erlaubt.

Daneben gibt es das Konzept der *Arbeitsgruppe*, damit Rechner auch logisch zusammengefasst werden können. Auch die Arbeitsgruppe hat einen NetBIOS-Namen.

Seit Windows 2000 kommt Microsoft aber auch nach und nach von NetBIOS ab. Zumindest in Domänenstrukturen mit einem zentralen Domänencontroller (der Begriff „Domäne“ wird hier etwas anders gebraucht), ist Microsoft auf DNS umgestiegen.

Im lokalen Netz wird aber auch unter Windows XP immer noch NetBIOS verwendet – auch die Netzwerkumgebung basiert auf NetBIOS. Daran wird sich wohl auch so schnell nichts ändern.

Tatsache ist, dass dein Windows-Client (so er mindestens Windows 2000 ist) mit seinem Rechnernamen sowohl einen DNS- als auch einen (gleichlautenden) NetBIOS-Namen hat.

## Wozu überhaupt Namen?

Was kannst du dir besser merken – `81.209.188.63` oder `www.knowware.de`? Hast du die IP-Adresse gewählt, melde dich bitte bei "Wetten dass …?" an, da gibt es für dich ein paar interessante Wetten zu gewinnen …

Bist du aber (so wie wir) eher der „normale“ Typ, wirst du dir die „sprechende“ Bezeichnung leichter merken. Eben darum hat man das Namenssystem erschaffen.

Jetzt braucht man nur noch jemanden, der einem sagt, welche IP-Adresse zu `www.knowware.de` gehört, da Computer ja nicht mit dem Namen selbst arbeiten können. Dazu gibt es zwei Möglichkeiten:

1. `/etc/hosts`: In dieser Datei finden wir eine lokale Zuordnung von Namen zu IP-Adressen.
2. *DNS-Server*: Du kannst deine Namensauflösung auch von dafür vorgesehenen Servern vornehmen lassen. Das Thema DNS ist nicht ganz ohne, daher werden wir dir das ab Seite 39 etwas genauer erläutern. Hier reicht es erst mal, wenn du weißt, dass jeder TCP/IP-fähige Rechner einen DNS-Client an Bord hat, der einen beliebigen DNS-Server für die Namensauflösung fragen kann. Die IP-Adresse dieses DNS-Servers – es können auch mehrere sein – trägst du dort ein, wo du auch den Namen eingabst:


Bei uns lassen wir diese Felder erst mal leer. Wir werden im Kapitel „Linux als DSL-Router“ auf Seite 36 ausführlich darauf zurückkommen. Also keine Panik, wenn du jetzt noch nicht den vollen Durchblick hast.

Zunächst wollen wir aber die erste Methode zur Namensauflösung verwenden. Schau dir unsere Datei `/etc/hosts` etwas näher an, indem du sie mit einem Editor öffnest – z.B. `vi`. Du findest hier unter anderem einen Eintrag in dieser Art:

```
192.168.0.2    tux.linux.local    tux
```

In der ersten Spalte steht die IP-Adresse; in der zweiten der volle Domain-Name (`tux.linux.local`). Die dritte Spalte ist optional und enthält einen Alias (`tux`).

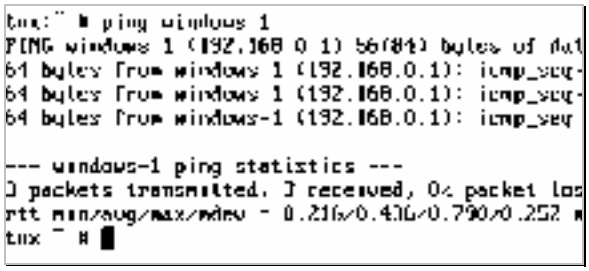
Setze unter diese Zeile eine weitere Namensauflösung:

```
192.168.0.1    windows-1    gulugulu
```

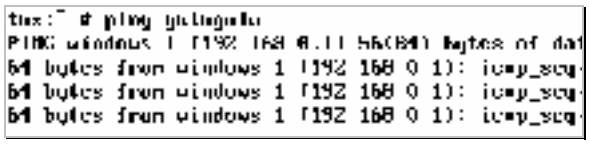
Hast du eine andere IP-Adresse für deinen Windows-Rechner gewählt, musst du diese eintragen. Speichere die Datei anschließend und beende den Editor.

*Und wozu soll das nun gut sein?*

Ab sofort kannst du den Windows-Rechner auch über seinen Host- bzw. DNS-Namen ansprechen. Gib zur Probe `ping windows-1` ein:




Wie du siehst, wird der Name zur IP-Adresse aufgelöst. Diese Namenseinträge haben nicht zwangsläufig etwas mit dem „echten“ Namen des Rechners zu tun – sie sind völlig willkürlich. Um dir das zu demonstrieren, haben wir als *Alias* in `/etc/hosts` den Namen `gulugulu` eingetragen. Funktioniert aber auch:




Unter Windows 2000/XP funktioniert das übrigens genauso. Die Datei befindet sich dort unter `c:\windows\system32\drivers\etc\hosts`.

Um die Sache noch mal klarzustellen:

Jeder Rechner, der TCP/IP spricht, hat zur lokalen Namensauflösung eine `hosts`-Datei. Die IP-Namenszuordnungen sind hier völlig willkürlich!

Natürlich macht es Sinn, hier die echten Namen einzutragen – alles andere ist Spielerei …

### 127.0.0.1 – localhost

Ist dir bei der Ausgabe des Befehls `ifconfig` der zweite Eintrag aufgefallen? Da stand was von `lo`!? In `/etc/hosts` findest du folgenden Eintrag:

```
127.0.0.1    localhost
```

Was hat es also mit diesem *localhost* auf sich? Ganz einfach: Jeder TCP/IP-fähige Rechner hat eine solche *loopback*-Adresse – eine Adresse, die immer auf den eigenen Rechner zeigt. Damit kann eine Netzwerkkomponente mittels *simulierter* Netzwerkkommunikation auf eine andere Komponente auf demselben Rechner zugreifen, ohne den Umweg über die Netzwerkkarte zu gehen. Später werden wir z.B. unseren Webserver durch Eingabe von `localhost` im Adressfeld des Browsers testen.

## Router – Die Pfadfinder

Wird ein Datenpaket an eine IP-Adresse außerhalb des eigenen Subnetzes geschickt, muss es über einen *Router* (auch *Gateway* genannt) gesendet werden. Dieser Router, sozusagen das Tor zur Außenwelt, hat eine Schnittstelle zu mindestens zwei Subnetzen (es können auch viel mehr sein). In kleineren Netzwerken haben Computer normalerweise nur einen Ausgang: das so genannte *Standardgateway*.

Klickst du im Dialogfenster KONFIGURATION DER NETZWERKKARTE auf ROUTING, kannst du die IP-Adresse des Standardgateways eintragen:


**Standardgateway – das Tor zur Außenwelt**

In unserem Fall ist das noch nicht nötig, da wir in unserer isolierten Laborumgebung keinen Weg nach draußen verwenden. Wir kommen aber im Kapitel *Linux als DSL-Router* ab Seite 36 ausführlich darauf zurück.

## … und noch mehr Background

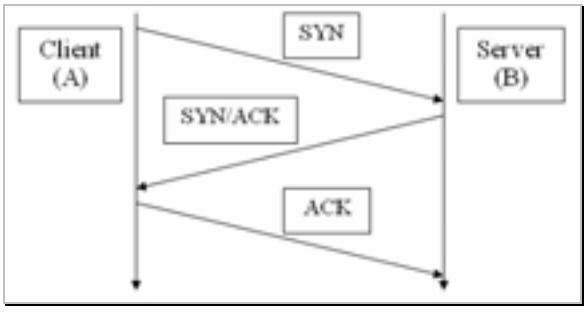
Das Folgende brauchst du zum Verständnis der späteren Kapitel – daher hier kurz ein paar Grundlagen:

### TCP – das Transportprotokoll

Auf IP setzt entweder TCP oder UDP als Transportprotokoll auf Netzwerkebene auf – sozusagen als Ladefläche unseres „Netzwerk-LKWs“, bei dem wir IP als die Räder definiert haben. Die meisten Anwendungen nutzen TCP, weil es sicherer ist. Warum das?

Nun – TCP baut, bevor die Datenübertragung selbst beginnt, eine „Sitzung“ zwischen den kommunizierenden Rechnern auf. In dieser Sitzung wird regelmäßig kontrolliert, ob alle Daten sauber angekommen sind. Die Eröffnung der Sitzung erfolgt über den so genannten *3-Way-Handshake*, das Drei-Wege-Händeschütteln – wir kannten bis dato nur Drei-Wege-Lautsprecherboxen …

Technisch läuft das so ab:




1. Will Rechner A mit Rechner B kommunizieren, schickt er ein Begrüßungspaket, in dem das so genannte SYN-Flag (ein „Flag“ ist eine Art Schalter) gesetzt ist. SYN steht für *Synchronisation*.
2. Die Antwort von B ist ein TCP-Paket, bei dem SYN und ACK gesetzt sind. Das ACK-Flag steht für *Acknowledge* (= Bestätigung).
3. Rechner A schickt nun noch eine Bestätigung seinerseits, ein Paket mit gesetztem ACK.

SYN, SYN/ACK und ACK könnte man also übersetzen:

A: Hey, B! Lass uns mal reden!

B: Okay, ich höre dir zu.

A: Fein, dann fang ich jetzt an.

Und schon startet die Gesprächsrunde.

Möchte B mit A *nicht* sprechen, sendet er als Antwort auf das erste SYN-Paket ein RST-Paket (für *Reset* = zurücksetzen). Eine laufende TCP-Kommunikation wird mit dem gesetztem FIN beendet (für *Final* = Schluss).

### Ports – Andockstationen für Clients

Will TCP eine Verbindung aufbauen, wird nicht einfach ein Paket von A an B versendet … Hä? Was denn noch?

TCP ist kein Selbstzweck; es ist ein Transportprotokoll für die Anwendungsprotokolle, die von den Netzwerkanwendungen verwendet werden. Jede Netzwerkanwendung bindet sich dazu an einen Port – eine Art Andockstation. Diese wird durch eine Zahl zwischen 1 und 65535 bezeichnet. Gängige Serveranwendungen haben genormte Ports – hier ein paar Beispiele:

| Anwendung | Protokoll | Port |
|---|---|---|
| Web (WWW) | HTTP | 80/tcp |
| E-Mail-Versand | SMTP | 25/tcp |
| Windows-Netzwerke | NetBIOS bzw. SMB und CIFS | 137/udp<br>138/udp<br>139/tcp<br>445/tcp |
| Namensauflösung (DNS) | DNS | 53/udp |

Will eine Clientanwendung auf A mit einem Server-Dienst auf B kommunizieren, klopft TCP an dem jeweiligen Port an. Nur wenn der Port auch durch den jeweiligen Dienst „gebunden“ ist, wird Rechner B mit einem SYN/ACK antworten; sonst kommt ein RST zurück.

Ein Beispiel: Nehmen wir an, auf B läuft ein Webserver. Dann sendet der *Webbrowser* auf A an B ein SYN-Paket auf Port 80. Der Server auf B antwortet mit SYN/ACK an den Port auf A, an den der Webbrowser gebunden ist.

Die Ports unter 1024 sind so genannte *Well known-Ports* und für Serveranwendungen reserviert. Die darüber liegenden Ports (bis 65353) sind für Netzwerk-Client-Anwendungen frei verfügbar und werden dynamisch vom Betriebssystem vergeben.

### UDP – Für den schnellen Austausch

Ist dir aufgefallen, dass wir hinter der Port-Nummer entweder `/tcp` oder `/udp` geschrieben haben? Das User Datagram Protocol (UDP) wird als Alternative zu TCP verwendet, wenn keine umfangreichen Daten ausgetauscht werden, sondern lediglich eine kurze Anfrage (z. B. nach einer Namensauflösung) erfolgt. UDP baut keine Sitzungen auf – geht ein Paket verloren, ist es eben weg. TCP dagegen hat Kontrollmechanismen, durch die der Datentransport zuverlässig sichergestellt ist.

Dafür ist UDP schneller, weil weniger Verwaltungsdaten im Protokollheader transportiert werden müssen.

Der *Header* eines Protokolls wird vor die Nutzdaten gesetzt; er enthält die Informationen, die für das jeweilige Protokoll wichtig sind – bei IP u.a. die *Quell-* und die *Zieladresse*, bei TCP und UDP die *Ports*. Bei TCP gibt es zusätzlich noch die besprochenen *Flags*.

Es gibt festgelegte Anwendungen, die UDP zur Datenübertragung nutzen; dazu gehören u.a.

- DNS (53/udp),
- Windows-Namensdienst (137/udp) und Windows-Datagramdienst (138/udp),
- NTP – Network Time Protocol (123/udp),

Um den Port eindeutig zu definieren, wird zusätzlich das entsprechende Transport-Protokoll angegeben (HTTP belegt dann z.B. Port 80/tcp).

Willst du mal sehen, welche Dienste es gibt, schau in die Datei `/etc/services` (Linux) bzw. `C:\WINDOWS\SYSTEM32\drivers\etc\services` (Windows).

# Samba – der Linux-Datei- und Druckserver für Windows

Zurück zu unserem Szenario – fassen wir die aktuelle Situation noch einmal zusammen:

- Anna hatte einen Plattencrash, dafür aber keine Datensicherung.
- Für die ganze Familie steht nur ein Drucker zur Verfügung, der am Rechner deines Mannes angeschlossen ist. Ist der ausgeschaltet, kann keiner drucken.
- Sollen Daten untereinander ausgetauscht werden, weiß keiner richtig, wie das zu bewerkstelligen ist.
- Öfters meldet sich ein Familienmitglied am Rechner eines anderen an, um z.B. „eben Mails zu checken". Dabei werden gerne Einstellungen geändert, sodass sich der „Besitzer" des Computers danach ärgert, weil seine schöne Konfiguration weg ist.

Was du brauchst, ist ein zentraler Datei- und Druckserver. Die perfekte Lösung hierfür ist *Samba*:

- Samba ist in fast jeder Linux-Distribution enthalten.
- Mit Samba wird jedem Nutzer ein eigenes *Home-Verzeichnis* übers Netzwerk zur Verfügung gestellt.
- Samba simuliert einen echten Windows-Server und integriert sich so nahtlos in deine Windows-Netzwerkumgebung.
- Du kannst alle wichtigen eigenen Dateien aller Benutzer zentral auf dem Samba-Server sichern.
- Der Austausch von Dateien wird vereinfacht, weil alle auf dieselbe Freigabe zugreifen.
- Samba dient auch als Druckserver – du musst den Drucker lediglich am Samba-Server anschließen und unter Linux konfigurieren.
- Samba kann noch mehr – z.B. eine zentrale Anmeldung und sogar Virenscanning.

So, jetzt haben wir aber genug aus dem Hochglanz-Prospekt vorgelesen … sehen wir uns lieber an, was Samba wirklich drauf hat.

## Samba installieren

Auf www.samba.org findest du die jeweils aktuelle Version. Andererseits stellt jede Distribution ein Samba-Paket zur Verfügung – so auch SuSE. Die z.Z. aktuelle Version ist Samba 3.0.11.

Der einfachste Weg ist, unter SOFTWARE|SOFTWARE INSTALLIEREN ODER LÖSCHEN in YaST nach dem Begriff *samba* zu suchen. Stelle sicher, dass folgende Pakete installiert werden, wenn noch nicht geschehen:

- `samba`
- `samba-client`
- `libsmbclient`
- `winbind`
- `samba-doc`

Die anderen gefundenen Pakete dürften schon *per Default* (in der Voreinstellung) installiert sein; es sind Hilfspakete für KDE und YaST.

Die restliche Prozedur bei der Installation kennst du, daher gehen wir jetzt nicht weiter darauf ein (klicke auf ÜBERNEHMEN, …). Anschließend sind die Samba-Pakete auf deinem System installiert.

## Den Samba-Daemon starten

Falls du dich jetzt über die Wortwahl wunderst, hier noch einmal zur Wiederholung:

Als *Daemon* werden Dienstprogramme bezeichnet, die als Hintergrundprozess laufen.

Samba besteht eigentlich aus zwei Diensten:

1. `nmbd` – stellt den Namensdienst zur Verfügung.
2. `smbd` – stellt den eigentlichen Samba-Service zur Verfügung.

Willst du wissen, ob diese Dienste gegenwärtig laufen, gibst du folgendes ein:

```
ps ax|grep mbd
```

Damit erfasst du beide Dienste in deinem `grep`-Filter. Im Moment dürfte aber noch nichts auftauchen – Samba läuft nicht. Das ändern wir jetzt mit folgenden Befehlen:

```
rcnmb start
rcsmb start
```

Samba sollte dies mit entsprechenden Erfolgsmeldungen quittieren. Ob die Dienste tatsächlich laufen, kannst du mit dem o.a. `grep`-Filter herausfinden.

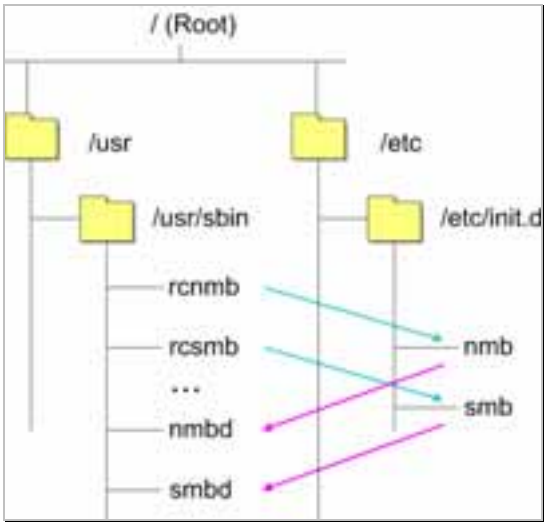
## Was steckt dahinter?

Hinter `rcnmb` bzw. `rcsmb` verbirgt sich das jeweilige Start-/Stop-Skript in `/etc/init.d` (auch *Init-Skript* genannt), wie du leicht selbst feststellen kannst:

`which rcnmb` zeigt dir, dass `rcnmb` in `/usr/sbin` liegt. Dort kannst du dir die beiden betreffenden Einträge anzeigen lassen:

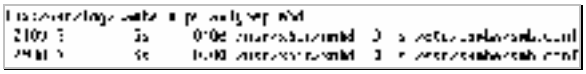
```
tux:/usr/sbin # ls -l rc*mb
lrwxrwxrwx  1 root root 15 Dec 23 13:40 rcnmb -> /etc/init.d/nmb
lrwxrwxrwx  1 root root 15 Dec 23 13:40 rcsmb -> /etc/init.d/smb
```

Wie du siehst, handelt es sich um *Softlinks* auf die Init-Skripte. Diese rufen übrigens wieder `nmbd` und `smbd` auf in `/usr/sbin`, also dem aktuellen Verzeichnis:




**Start von Samba über rc-Softlinks**

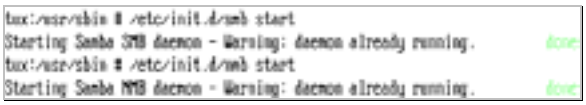
Genau diese tauchen dann auch in der Prozess-Liste auf:



Diese `rc`-Softlinks gibt es nur bei SuSE. Andere Distributionen regeln dies anders – ein Start-/Stop-Skript unter `/etc/init.d` oder `/etc/rc.d/init.d` existiert aber fast immer. Dieses kannst du natürlich über folgende Befehlszeilen auch direkt ansprechen:

```
/etc/init.d/nmb start
/etc/init.d/smb start
```

Wenn du das jetzt versuchst, bekommst du eine Warnmeldung, da Samba ja schon gestartet wurde.




Andere mögliche Parameter für die Init-Skripte sind:

- `stop` (stoppt den Service)
- `restart` (dasselbe wie `stop` + `start`)
- `reload` (nur für `smb` – liest die Konfiguration neu ein)
- `status` (überprüft selbigen)

## Automatischer Start von Samba

Eleganter wäre es natürlich, wenn wir Samba über die Runlevel-Verzeichnisse einbinden könnten, damit es automatisch gestartet wird, wenn der Rechner hochgefahren wird. Dazu gehst du in den *Runlevel-Editor* über YAST2|SYSTEM|RUNLEVEL-EDITOR und trägst für `nmb` und `smb` die Runlevel 3 und 5 ein:

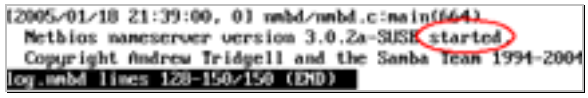

Wechsle dazu in den EXPERTENMODUS und markiere den entsprechenden Dienst. Unten kannst du nun Runlevel 3 und 5 anklicken.

Natürlich kannst du auch manuell entsprechende Softlinks in den Runlevel-Verzeichnissen erstellen; wie, das siehst du im Heft „Linux für Poweruser“ ab Seite 44.

## Den Dienst testen

*Ein laufender Daemon macht noch keinen funktionierenden Netzwerkdienst*. In diesem Sinne sehen wir uns jetzt an, was wir über *Samba* herausfinden können.

Zunächst schauen wir in die Logdateien. Diese befinden sich unter `/var/log/samba` und heißen `log.nmb` bzw. `log.smb`. Hier werden Fehler- und Statusmeldungen festgehalten:




**Der nmb-Dienst läuft …**

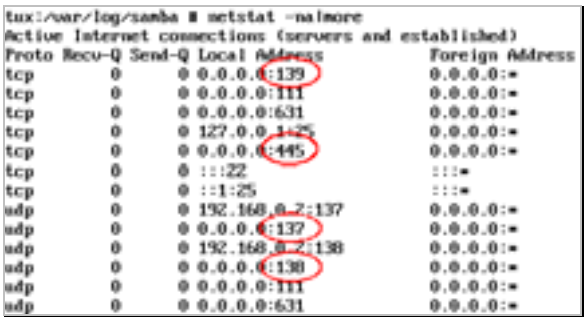
Als Nächstes überprüfen wir den Netzwerkstatus:

Läuft *Samba* (also `nmbd` und `smbd`), lauscht der Service auf den Ports 137/udp, 138/udp und 139/tcp sowie

445/tcp. Diese müssen demnach in der Liste der aktiven Ports deines Servers auftauchen. Das überprüfst du mit folgendem Befehl:

```
netstat -na|more
```

Gleich zu Beginn findest du die entsprechenden Einträge unter der Rubrik *Active Internet Connections*.




**Samba läuft, die Ports sind "gebunden"**

`netstat` ist, der Name sagt es, ein multifunktionales Tool zur Anzeige des `Netzwerkstatus`. Die wichtigsten Optionen sind:

| Option | Beschreibung |
|---|---|
| a | Alles anzeigen |
| n | Numerische Portbezeichnungen |
| r | Routing-Tabelle |
| v | Ausführlich (verbose) |

## Den Samba-Dienst testen

Du hast alles überprüft, und bisher sieht es gut aus? Nun wollen wir sehen, ob *Samba* tatsächlich funktioniert. Zwar ist *Samba* eigentlich ein Windows-Server, aber du kannst den Dienst auch prima unter Linux selbst testen.

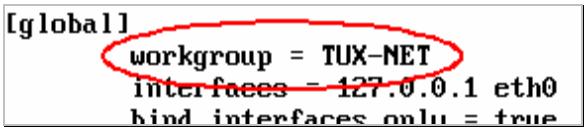
Dazu bringt *Samba* eine Reihe von Tools mit, die wir gleich vorstellen wollen. Jetzt aber klären wir erstmal …

### Die Arbeitsgruppe

Zunächst werfen wir einen Blick auf die (einzige) Konfigurationsdatei von *Samba*. Sie heißt `smb.conf` und befindet sich meistens unter `/etc/samba`:

```
less /etc/samba/smb.conf
```

Uns interessiert hier nur der `workgroup`-Eintrag gleich oben unter der Sektion `[global]`:




Dies unterscheidet sich je nach *Samba*-Version. Da der Samba-Server den Namen des Linux-Hosts übernimmt, wissen wir über unseren *Samba*-Server folgendes:

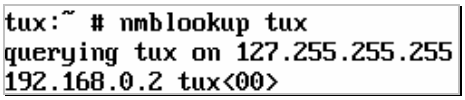
NetBIOS-Name: TUX

Arbeitsgruppen-Name: TUX-NET

Diese beiden Informationen werden wir nun schamlos ausnutzen ;-).

### nmblookup

Dieses Tool löst die NetBIOS-Namen auf. Zuerst testen wir den Server-Namen. Groß- und Kleinschreibung wird von NetBIOS ignoriert:

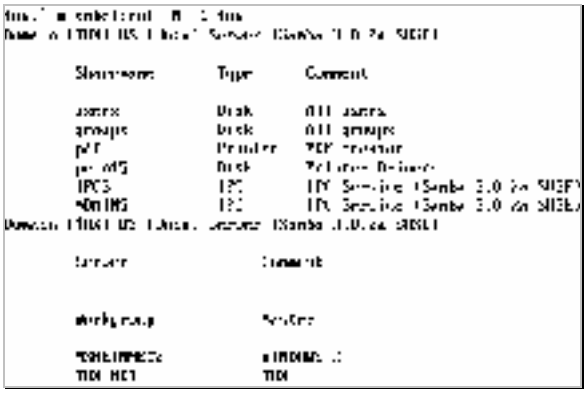



Sieht gut aus. Reagiert er auch auf die Arbeitsgruppe?

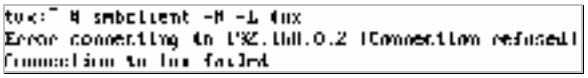

Prima. Diese merkwürdige IP-Adresse 127.255.255.255 ist übrigens die *Broadcast-Adresse* des *Loopback*-Subnetzes 127.0.0.0/8 – (siehe Seite 11) - zu deutsch: Er brüllt sich selbst an ...

### smbclient

Dieses Tool testet die eigentliche Funktion von *Samba*: Zugriffe auf Freigaben, Drucker, etc. Wir geben hier `smbclient -N -L tux` ein:



Na sieht doch bestens aus … auch wenn wir die Zeilen vielleicht noch nicht im Einzelnen deuten können. Wenn Samba *nicht* läuft, zeigt sich das so:



`smbclient` kennt eine ganze Reihe von Optionen. Die wichtigsten sind:

| Option | Bedeutung |
|---|---|
| -N | keine Passwort-Abfrage |
| -L <Server> | Zeigt alle Freigaben des angegebenen Samba-Servers |
| -U <User> | Benutzer für die Anmeldung |

Wie du siehst, kannst du auch ganz ohne Windows-Client deinen Samba-Server benutzen bzw. zumindest austesten. Natürlich macht das nicht so richtig Spaß – außerdem geht es doch wohl etwas am Ziel vorbei, einen zentralen Datei- und Druckserver für das Windows-Netzwerk aufzusetzen. Also werden wir nun mit der Konfiguration unseres „südamerikanischen Tanzes“ beginnen.

# Samba Grundkonfiguration

Als erstes machen wir *Tabula Rasa*: Taufe `smb.conf` um in `smb.conf.old` und erstelle `smb.conf` neu als leere Datei. Wir werden die Konfiguration von Samba von Grund auf selbst schreiben.

## Die Sektion [global]

Bei einem ersten Blick auf die ursprüngliche `smb.conf` hast du es vielleicht schon bemerkt: Die Konfiguration ist in Sektionen aufgeteilt, die in eckigen Klammern stehen. `[global]` ist die erste Sektion. Hier werden die Rahmenbedingungen definiert.

Samba kennt über 300 Konfigurations-Parameter. Viele haben einen Default-Wert und müssen in `smb.conf` nur aufgeführt werden, wenn ihr Wert angepasst werden soll. Soll heißen: Wir müssen nur wenige Parameter definieren. Los geht's: Schreibe Folgendes in die Datei und rücke die Parameterzeile durch Tabulator ein:

```
[global]
        workgroup = sambanet
```

Fertig! Zu einfach? Vielleicht! Aber es funktioniert! Du glaubst uns nicht? Wir beweisen es dir: Speichere deine neue `smb.conf` und gib `testparm` ein. Damit testest du die Konfigurationsdatei von Samba auf Fehler:

```
tux:/etc/samba # testparm
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

# Global parameters
[global]
        workgroup = SAMBANET
```

Es werden keine Fehler gefunden – alles fein! Das Tool `testparm` solltest du immer aufrufen, wenn du eine Änderung an `smb.conf` vorgenommen hast. Anschließend musst du beide Dienste (`nmb` und `smb`) neu starten, damit die Änderungen wirksam werden:

```
rcsmb restart
rcnmb restart
```

oder

```
/etc/init.d/smb restart
/etc/init.d/nmb restart
```

Ab sofort ist unser Samba-Server Mitglied der Arbeitsgruppe SAMBANET. Mehr kann er aber noch nicht. Wir brauchen eine Freigabe!

## Die erste Freigabe

Jede Freigabe (engl. *Share*) hat eine eigene Sektion in `smb.conf`. Bevor wir jedoch eine solche Sektion erstellen, müssen wir festlegen, wie Benutzer authentifiziert werden, wenn sie auf Ressourcen des Samba-Servers zugreifen. Das kann unterschiedlich erfolgen.

Der Parameter, der dies systemweit regelt, heißt `security`; er kann insbesondere folgende Werte annehmen:

1. *share*: Authentifikation auf Freigabe-Ebene, d.h. die Freigaben sind de facto ungeschützt. Unter Windows ist es möglich, ein Passwort zu vergeben. Wer das Passwort kennt, darf rein. Ist kein Passwort gesetzt, gibt es keine Passwort-Abfrage, und jeder hat Zugriff. Unter *Samba* funktioniert das ausschließlich *ohne* Passwort!
2. *user*: Authentifikation auf Benutzerebene: Der Benutzer muss sich mit einer Benutzerkennung authentifizieren, die auf dem Samba-Server selbst bekannt ist.

Wir werden mit einer einfachen Freigabe ohne Passwort anfangen, also ergänzen wir die Sektion `[global]` um die folgende Zeile:

```
        security = share
```

Nun wird es Zeit, eine Testfreigabe zu erstellen. Dazu brauchen wir ein Verzeichnis auf dem Linux-Server:

```
mkdir /sambatest
```

Anschließend erstellen wir die neue Sektion in `smb.conf`. Diese sieht dann so aus:

```
[global]
        workgroup = sambanet
        security = share


[test]
        comment = Testfreigabe
        path = /sambatest
        read only = no
        guest ok = yes
```

Hier die Erläuterung der Parameter:

| Parameter | Bedeutung |
|---|---|
| `comment` | Kommentar, erscheint auch in der Netzwerkumgebung |
| `path` | gibt das lokale Verzeichnis an, das freigegeben werden soll |
| `read only` | `yes` = nur lesen<br>`no` = lesen + schreiben |
| `guest ok` | `yes` = keine Passwortabfrage – muss so gesetzt sein bei `securtiy = share` |

Die Freigabe heißt übrigens immer so wie der Sektionsname: `test`.