Linuxeinsteiger.org

-

Das Buch zur Seite

Dresden im Juni 2002

Vorwort

Nun halten Sie die erste Ausgabe des Linuxeinsteiger.org - Buches in Händen, aber das ist nur der Anfang, denn meiner Meinung nach wird das Buch nie fertig sein, denn es werden immer wieder neue Anleitungen und Hilfestellungen dazu kommen, die den User ein wenig weiter bringen.

Diese Ausgabe ist natürlich vom Umfang nicht sonderlich ergiebig, aber es sollte damit doch schon möglich sein, dass einem wirklichen Newbie, der sich noch nie mit Linux beschäftigt hat, einige Fragen beantwortet werden können. Und das ist auch unser Ziel.

Natürlich verfolgen wir mit diesem Buch keinerlei Gewinnerzielungsabsichten, so dass der Download und das Weiterverbreiten gestattet ist, jedoch behalten wir uns alle Rechte vor.
Aufgrund dessen ist und bleibt das Buch kostenlos. Sollte sich irgendwann einmal ein Verlag finden (was absolute Zukunftsmusik wäre), der dieses Buch verlegen würde, so wäre das Buch auf jeden Fall auch weiterhin kostenlos als Downloadversion erhältlich.

Bisher haben sich an diesem Werk 3 Personen - 2 daran aktiv - beteiligt. Natürlich wären wir sehr froh, wenn sich der ein oder andere User an diesem Projekt beteiligen würde, denn meist entdeckt man doch Kleinigkeiten im Linuxalltag über die man berichten kann.

Deshalb richtet bittet Eure Artikel (und Spenden *g) an

webmaster@linuxeinsteiger.org .

Und nun viel Spaß an der Lektüre ... Möge das Wissen mit Euch sein!

Euer Linuxeinsteiger.org - Team!

Inhaltsverzeichnis

1.	Systemkonfiguration	
1.1	Immer die richtige Zeit durch Verwendung eines Zeitservers [3]	4
1.2	Cron - der stille Helfer [3]	5
1.3	SCSI - Emulation eines CD - Brenners [3]	7
1.4	Einfach und sicher einen neuen Kernel kompilieren [2,3]	7
2.	Netzwerk	
2.1	Konfiguration eines Routers [3]	12
2.2	So realisiere ich erfolgreich ein echtes Netzwerk unter Linux [1]	14
2.3	Firewalling mit iptables [3]	17
3.	Tipps und Tricks	
3.1	PDF - Dateien erstellen [3]	21
3.2	Benutzerrechte editieren [1]	22
3.3	Linken - aber richtig! [1]	24
3.4	Manuals - Oder wie helfe ich mir selbst? [1]	25
3.5	X - Window - Server - Session freigeben [3]	26
4.	Workshops	
4.1	Dateien und Verzeichnisse [3]	
4.1.1	Teil 1	27
4.1.2	Teil 2	31
4.1.3	Teil 3	34
5.	Sonstiges	
5.1	Edonkey - Installationsanleitung [3]	37
6.	Autorenverzeichnis	39

1. Systemkonfiguration

In diesem Kapitel beschäftigen wir uns mit den administrativen Aufgaben, die ein jeder Linux - User durchführen sollte, damit man entweder das System vernünftig benutzen kann, bzw. um sich einige Arbeiten zu erleichtern.
Diese Dinge können meist nur vom Superuser Root durchgeführt werden, sollten aber inhaltlich nicht ignoriert werden, falls man am System keine Root - Rechte hat, denn man weiß ja nie, ob man dieses Wissen nicht doch einmal einsetzten könnte.

1.1 Immer die richtige Zeit durch Verwendung eines Zeitservers

Da kauft man sich einen neuen Rechner für 1000.-- € - oder mehr - und die Uhr läuft nach spätestens einem Monat falsch. Wer es leid hat, die Uhrzeit immer wieder per Hand zu stellen, dem kann diese kleine Anleitung Abhilfe verschaffen.

Prinzipiell kann die Systemzeit direkt in der Konsole mit dem Befehl
`# netdate ptbtime1.ptb.de && clock -w`
sekundengenau mittels eines Zeitservers gestellt werden, wobei darauf zu achten ist, dass sich jener Befehl nur als Root ausführen läßt.

Anmerkung: Das Paket „netdate“ muss natürlich vorher installiert worden sein. Deshalb ist es ratsam, obigen Befehl vorher per Konsole auszuführen, bevor man dann das Script implementiert.

Eleganter läßt sich der Vorgang allerdings mit Hilfe eines Scipts lösen. Hierfür erstellen wir eine Datei (beispielsweise in /usr/local/bin) namens timeset, wobei Root der Eigentümer der Datei sein sollte und die Datei mit den richtigen Rechten ausgestattet sein sollte (755). Nun editieren wir die Datei folgendermaßen:

```
#!/bin/sh
#
#Uhr beim Starten der Internetverbindung stellen
#
netdate ptbtime1.ptb.de  &&  clock -w
```

und speichern diese.

Abschließend müssen wir nur noch am Ende der Datei /etc/ppp/ip-up zum obigen Script verlinken, in dem wir am Ende /usr/local/bin/timeset vermerken.

Somit wird bei jedem Start der Internetverbindung die Systemzeit per Zeitserver wieder exakt gestellt.

1.2 Cron - der stille Helfer

Wer steht nicht öfters vor dem Problem, dass gewisse Dinge immer zur selben Zeit oder an einem bestimmten Tag ausgeführt werden möchten?
Ich denke da beispielsweise an wöchentliche Backups, an Skripte, die immer wieder zu einer bestimmten Zeit ausgeführt werden sollen, etc. Und was macht der unwissende User?
Er führt diese Dinge immer wieder manuell aus und dann werden diese lästigen Routinearbeiten irgendwann vergessen...
Aber ab heute ist endgültig Schluss damit, denn wir lernen nun den Umgang mit dem Cron - Dämon. Dieser fragt jede Minute, ob für ihn Arbeit vorliegt und führt diese dann auch noch aus. Unter dem Strich handelt es sich also um einen neuen Mitarbeiter, der für Euch auch noch kostenlos die ungeliebte Arbeit erledigt. Feine Sache, oder?
Jetzt fehlt uns nur noch die Funktionen von Cron zu verstehen und dann haben wir ein mächtiges Hilfsprogramm.

Also was macht Cron? Naja, nichts weiter als eine angelegte Tabelle (crontab) aus einer Datei auszulesen. Klingt einfach und ist auch so!
Und so ist die Tabelle grundsätzlich aufgebaut:

Minuten	Stunden	Tage	Monate	Wochentage	User	Befehl
0 - 59	0 - 23	' 1 - 31	' 1 - 12	0 - 7		

Jetzt noch zu den Besonderheiten und dann können wir schon loslegen.
Einerseits wird ein kompletter Bereich durch ein * gekennzeichnet, was in der Spalte Minuten eben zu jeder Minute bedeuten würde. Ferner erkennt man, dass die Wochentage von 0 - 7 gehen, was also 8 Tagen entsprechen würde. Hier ist der Sonntag dem Eintrag Null oder Sieben zuzuordnen.
Des weiteren kann man innerhalb einer Spalte mehrere Eingaben mit einem Komma voneinander trennen, Bereiche mit Hilfe eines Bindestriches und Intervalle durch einen Schrägstrich definieren.

So viel zur Theorie, nun zur Praxis...

Zuerst wollen wir uns über die generellen Einträge als Root unterhalten. Um als Root einen Cronjob anzulegen, editieren wir mit einem Texteditor unserer Wahl die Datei /etc/crontab nach dem angesprochenen Schema.
Hierzu wollen wir ein paar beispielhafte Einträge besprechen, um die Funktionsweise von Cron endgültig zu verstehen.... Die /etc/crontab könnte dann so aussehen:

```
SHELL=/bin/sh
PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin
MAILTO=root
#
# check scripts in cron.hourly, cron.daily, cron.weekly, and cron.monthly
#
#Jeden Tag, um 0:00 Uhr wird das Skript Backup ausgeführt
0  0  *   *  *   root  /usr/local/backup.sh
#Alle 2 Stunden wird eine Minute nach der vollen Stunde das Programm
#Sitecopy für den Benutzer nono ausgeführt
1  */2  *  *  *    root su -c "sitecopy -u linuxeinsteiger" nono
#Alle 15 Minuten zwischen 12:00 Uhr und 24:00 Uhr an den Wochentagen
#Dienstag und Donnerstag wird das Backup - Skript ausgeführt
*/15  12-0 *  *  2,4  root /usr/local/backup.sh
#
#ENDE
```

Wie wir nun gesehen haben, ist der User in der /etc/crontab immer Root... Natürlich können wir Programme wie im 2. Beispiel auch für andere User ausführen lassen, aber was machen wir, wenn man an einem System sitzt, an dem man eben keine Root - Rechte hat?
Natürlich ahnen Sie schon was kommt, denn auch hierfür hat Linux wieder die richtige Lösung parat, denn man kann auch als User Cronjobs anlegen, aber eben nicht in der globalen /etc/crontab , sondern unter dem jeweiligen Useraccount.

Die Eingabe von # crontab -l zeigt die Cronjobs des jeweiligen Users, # crontab -r löscht alle Einträge und mit # crontab -e werden diese bearbeitet.
Zweierlei Dinge sind hier noch wichtig zu erwähnen:

1. Die 6. Spalte "User" fällt logischerweise weg.... Also besteht die Crontab des Users grundsätzlich nur aus 6 Spalten!
2. Beim Bearbeiten der Beiträge öffnet sich der etwas gewöhnungsbedürftige Editor "vi"...

Durch Eingabe von [i] lässt sich die Crontab bearbeiten, gespeichert wird mit [:][w] , nachdem man den Edit-Modus mit [ESC] verlassen hat, und durch [:][q] beendet man den Editor.

Und nun viel Spaß, denn ab jetzt arbeitet euer Computer für Euch....

1.3 SCSI - Emulation eines CD - Brenners

Um unter Linux einen ATAPI - Brenner nutzen zu können, muss er als SCSI - Laufwerk emuliert werden. Dies geschieht durch Editieren einiger Dateien, was in den folgenden Schritten erklärt wird. Als Referenz diente SuSE Linux 7.3 Pro, jedoch müsste die Prozedur mit allen anderen Distributionen annähernd gleich funktionieren.

In der */ect/lilo.conf* ist über */boot=/dev/hda* folgendes einzutragen: **append="hd?=ide-scsi"** , wobei das Fragezeichen für den belegten IDE - Port steht.... (a = Primary Master, ..., d = Secondary Slave)

 # lilo in der Konsole eingeben, damit die Änderungen in den MBR der Festplatte eingetragen werden.

Die Datei */ect/modules.conf* öffnen und den Eintrag *alias scsi_hostadapter off* in **alias scsi_hostadapter ide-scsi** ändern.

Zum Schluss wird die Datei */etc/init.d/boot.local* geöffnet und es wird ganz unten die Zeile **/sbin/modprobe ide-scsi** ergänzt.

Nun wird durch Eingabe von *# init 6* der Rechner neu gestartet, wobei beim Startvorgang der Brenner als SCSI - Laufwerk aufgeführt sein sollte.
Abschließend wird in der */ect/fstab* dem Laufwerk der richtige Device zugeordnet. Deshalb ist der Eintrag **/dev/hd? /cdrecorder** in **/dev/scd0 /cdrecorder** zu ändern.
Mit dem Befehl *# cdrecord --scanbus* läßt sich überprüfen, ob die Emulation erfolgreich war und
somit ist die SCSI - Emulation abgeschlossen und dem Brennen unter Linux steht nichts mehr im Wege.
Viel Spaß damit!

1.4 Einfach und sicher einen neuen Kernel kompilieren

Jeder Einsteiger und noch nicht so erfahrene Linux - User stand mit Sicherheit schon einmal vor der Versuchung, einen eigenen Kernel zu kompilieren. Wichtig ist, dass der User sich für dieses Vorhaben

richtig viel Zeit schafft, denn je nach Prozessor- und RAM - Leistung kann das eigentliche Kompilieren zwischen 30 Minuten und mehreren Stunden in Anspruch nehmen.
Zwar findet man in Büchern und im Internet immer wieder ellenlange Anleitungen, aber meist weichen diese doch stark voneinander ab und der Anwender verliert relativ schnell das Interesse an dieser Materie, da er von den vorliegende Informationen fasst erschlagen wird.
Mein Ziel ist ein anderes, denn ich möchte mit dieser Anleitung einen Workshop kreieren, der für jeden leicht verständlich und nachvollziehbar ist. Als Referenzsystem fungiert SuSE Linux 7.3 Pro, jedoch sollte dieser Workshop auch für alle anderen Distributionen anwendbar sein.

Im folgenden wollen wir Schritt für Schritt den neuen Kernel kompilieren, jedoch ist es nötig, dass zuerst die neuen Kernel - Quellen von http://www.kernel.org besorgt werden. Ich empfehle den Download des aktuellen tar.gz - Archivs, wobei es sich empfiehlt, dass eine stabile Version (siehe 4) verwendet wird, da Entwicklerkernel sich für den alltäglichen Einsatz nicht eignen.

Nun zur Anleitung:

1) Wir loggen uns mit # su - als Root ein.
2) Wir wechseln in das Verzeichnis /usr/src und sehen nach, ob ein Verzeichnis namens linux vorhanden ist, also

   # cd /usr/src

   # ls -al

   Ist diese Verzeichnis nicht existent, dann geht es weiter mit 4. , sonst mit 3.
3) Da auf das Verzeichnis ein symbolischer Link gesetzt ist und wir diesen Link aber auf den neuen Kernel setzen müssen, gilt es dieses „leere“ Verzeichnis zu löschen. Bei mir sieht der Link bisher so aus:

   lrwxrwxrwx 1 root root 12 Apr 17 17:46 linux -> linux-2.4.18

   Demnach verweist das Verzeichnis linux auf den aktuell installierten Kernel 2.4.18. Nun löschen wir den Link mit

   # rm linux
4) Nun packen wir mit dem Befehl

   # tar xfvz /Pfad/wo/die/Quellen/liegen/linux-2.5.8.tar.gz

   die Quellen des neuen Kernel aus. Die Versionsnummer kann natürlich abweichen und hängt vom jeweiligen Kernel ab, der zu kompilieren ist. Ergänzen möchte ich nur, dass es sich in diesem Fall um eine Entwicklerversion handelt, da die Zahl an zweiter Stelle ungerade ist. Stabile Versionen haben immer gerade Zahlen an jener Stelle.

   Nun gibt es wiederum 2 Möglichkeiten. Entweder uns liegt nun ein Verzeichnis „linux“ vor, dann geht es weiter mit 5. , falls nicht dann überspringen wir diesen Schritt.
5) Wir benennen das Verzeichnis „linux“ in „linux-2.5.8“, bzw. in die jeweilige Version, die zu kompilieren

ist, also

mv linux linux-2.5.8

6) Jetzt setzen wir mittels

ln -s linux-2.5.8 linux

einen symbolischen Link vom Verzeichnis „linux-2.5.8“ auf das Verzeichnis „linux“.

7) Anschließend überprüfen wir die minimalen Systemanforderungen indem wir mit

cat /usr/src/linux/Documentation/Changes

die entsprechende Datei aufrufen und diese, sowie die notwendigen Befehle zum Überprüfen, unter „Current Minimal Requirements“ zu finden sind. Falls alle Anforderungen erfüllt sind, so kann es getrost weiter gehen, ansonsten sind diese per Installation zu erfüllen.

8) Im nächsten Schritt gehen wir mit

cd /usr/src/linux

in jenen Ordner und rufen mit

ls -al

den Inhalt des Verzeichnises auf. An dieser Stelle angelangt lautet die Frage, ob eine Datei namens .config vorhanden ist, da diese die notwendige Konfiguration des Systems enthält.

Nun bieten sich 3 Möglickeiten:

(a) Wir übernehmen die alte Konfiguration, dann geht es mit Punkt 9 weiter

(b) Wir nehmen die alte Konfiguration als Ausgangspunkt für eine Neukonfiguration, dann geht es logischerweise ebenfalls mit dem Punkt 9 weiter

(c) Oder wir erstellen eine eigene .config, dann würde es mit Punkt 11 weitergehen

9) Um die alte Konfiguration zu übernehmen, oder diese als Clone weiter zuverwenden, benötigen wir eine alte .config. Diese erhalten wir beispielsweise, wenn wir die Kernelquellen unseres Systems nachinstallieren, falls diese nicht schon vorhanden sind.

Unter SuSE - Linux finden wir diese im Verzeichnis /usr/scr/linux-2.4.10.SuSE . Wir wechseln in jenes Verzeichnis und kopieren die .config (erscheint mit dem Befehl # ls -al) in das Verzeichnis „linux“, also

cd /usr/src/linux-2.4.10.SuSE

cp .config /usr/src/linux

10) Jetzt wechseln wir in die Root - Bash (System --> Kommandozeilen) [nur nötig, falls wir unter X die Kernelkonfiguration vornehmen wollen] und führen im Verzeichnis /usr/src/linux den Befehl

make oldconfig

aus. Ab jetzt wird eine Vielzahl von Fragen erscheinen, die man alle mit N (NO) beantworten kann, falls am Ende immer NEW steht. Mit # ls - al stellen wir im nachhinein fest, dass nun eine .config und eine .config.old vorhanden sind. Die .config.old können wir getrost mit

rm .config.old

löschen.

11) Diejenigen, die einfach die alten Einstellungen übernehmen wollen, können diesen Schritt nun einfach überspringen. Für alle anderen ist dies die Möglichkeit, den Kernel an die eigenen Bedürfnisse anzupassen. Dazu stehen uns 3 Möglichkeiten zur Verfügung:
   (a) entweder die graphische Konfiguration, dann führen wir # make xconfig aus
   (b) oder textbasiert und bunt, dann geben wir # make menuconfig ein
   (c) oder für die Hardcores unter uns # make config
   Viel mehr kann man zur Konfiguration nicht sagen, denn das hängt schließlich von den eigenen Bedürfnissen und der verwendeten Hardware ab. Wir beenden die Konfiguration mit save & quit.

12) An dieser Stelle kommt jetzt endlich der eigentliche Kompiliervorgang, der aus mehreren Schritten besteht, jedoch zu einem Befehl zusammengefaßt werden sollte, da man so nicht die ganze Zeit auf den Monitor starren muss. Also geben wir folgendes ein:
   # make dep && make bzImage && make modules && make modules_install
   Das ganze Prozedere wird, wie schon oben angesprochen, eine Weile dauern und man kann sich in dieser Zeit erst einmal in aller Ruhe zurücklehnen oder etwas anderem widmen. Nebenbei erwähnt werden die Befehle nacheinander abgearbeitet, wofür die Zeichenfolge && verantwortlich ist.

13) Langsam nähern wir uns dem Ende der Installation, wobei wir in diesem Schritt mit dem Editor unserer Wahl (ich verwende meist den Midnight Commander) die Datei /etc/lilo.conf aufrufen, die folgendermaßen aussehen müßte: (ohne das blau unterlegte !)

```
(...)
image  = /boot/vmlinuz.xxx
label  = linux.xxx
root   = /dev/hda7
initrd = /boot/initrd.xxx
append = "enableapic vga=0x0317 hdc=ide-scsi"
(...)
```

   Wir kopieren nun jenen Inhalt ans Ende der Datei und editieren beide Einträge, wobei der blau markierte Teil relevant ist. Im obigen Eintrag ersetzen wird .xxx mit .org und im unteren Eintrag mit .own . Nachdem wir dies erledigt haben, speichern wir die Datei.

14) Im weiteren wechseln wir mit # cd /boot in jenes Verzeichnis und führen den Befehl # ls aus. Wie man erkennt, befindet sich eine Datei namens „vmlinuz“ in diesem Verzeichnis. Dies ist der Orginalkernel, welchen wir gemäß der obigen Änderungen an der lilo.conf anpassen. Deshalb führen wir die Befehle
   # mv vmlinuz vmlinuz.org
   #mv initrd initrd.org

aus, da auch die vorhandene „initrd“ den obigen Spezifikationen angepasst werden muss.
Falls eine Datei „System.map“ vorliegt, verfahren wir mit dieser genauso.
Dieser wichtige Schritt hat es uns ermöglicht, dass wir vom alten Kernel ein Backup erstellten, auf das wir wieder zurückgreifen könnten, falls der neue Kernel nicht fehlerfrei bootet (siehe 18)

15) Jetzt wechseln wir wieder in das Verzeichnis „linux“ mit # cd /usr/src/linux , und kopieren die dort vorhandene Datei „System.map“ nach /boot, also
cp System.map /boot

16) Im Anschluß daran wechseln wir mit # cd /usr/src/linux/arch/i386/boot in jenes Verzeichnis und überprüfen, ob sich dort die Datei „bzImage“ befindet. Falls dem so ist und das Datum auch stimmt, dann HERZLICHEN GLÜCKWUNSCH, denn das ist der neue Kernel. Diesen kopieren wir jetzt mit
cp bzImage /boot/vmlinuz.own
ins Verzeichnis /boot und benennen ihn bei dieser Gelegenheit auch gleich in „vmlinuz.own“ um.

17) Nun sind wir nur noch einen Steinwurf vom Ende entfernt. Wir wechseln mit # cd /.. ins Wurzelverzeichnis , führen die Befehle
mk_initrd -k vmlinuz.own -i initrd.own
lilo
aus. Wurde alles fehlerfrei hinzugefügt, so ist die eigentliche Installation abgeschlossen.

18) Nach einem Neustart des Rechners wählen wir im Bootmanager den neuen Kernel aus und geben init 3 als Parameter mit. Sollte der neue Kernel problemlos booten, so kann man ihn im nächsten Schritt als Standardkernel einrichten.

19) Hierfür rufen wir noch einmal die /etc/lilo.conf auf, vertauschen .org und .own (siehe Pkt. 13) und führen # lilo aus.

Damit ist der neue Kernel voll einsetzbar. Falls Probleme während der Prozedur auftreten, so stehen wir Ihnen hilfreich zur Seite. Viel Spaß!

2. Netzwerk

Nun wollen wir uns einer der Paradedisziplinen von Linux zuwenden, nämlich der ausgesprochenen Stabilität und Funktionsvielfalt innerhalb von Netzwerken. Dies gilt sowohl für homogene, als auch für heterogene Netzwerke.
Lassen wir uns also überraschen und inspirieren von dem, was uns Linux bieten kann.

2.1 Konfiguration eines Routers

Mit zunehmender Verwendung von DSL - Anschlüssen und der Tatsache, dass der Trend zum Zweitrechner - wie einem Laptop - geht, stellt sich dem Anwender die Frage, wie sich beide Rechner die Internetverbindung teilen könnten.
Die Lösung des Problems ist denkbar einfach und wir mittels Routing gelöst.

Folgende Voraussetzungen müssen erfüllt sein:

1. Der Client muss über eine richtig konfigurierte Netzwerkkarte verfügen. Hier könnte man als IP-Adresse 192.168.0.2 wählen.
2. Der Router muss über zwei Netzwerkkarten verfügen, wobei diese ebenfalls richtig konfiguriert sein müssen. Als IP - Adressen bieten sich für den DSL - Anschluß 192.168.22.1 (Dummy-IP) und 192.168.0.1 an.
3. Router und Client müssen per Netzwerkkarten vernetzt sein und der Router muss zudem eine korrekt eingerichtete DSL - Verbindung aufweisen. Dial - on - Demand wird empfohlen.
4. Da die Anleitung auf *iptables* basiert, ist ein Kernel ab Version 2.4 ebenfalls nötig.

Zuerst wollen wir den Router konfigurieren. Hierzu wechseln wir mit # cd /etc/ppp in jenes Verzeichnis und erstellen eine Datei namens "routing“ (# touch routing), die anschließend mit # chmod 755 routing ausführbar gemacht wird.
Nun wird folgendes Script in der Datei "routing“ erstellt:

```
#!/bin/sh
#
#Starten des Routers und des Forwardings
#
echo 1 > /proc/sys/ipv4/ip_forward
#Bei Verwendung ab SuSE 8.0 bitte anstatt obiger Zeile folgendes schreiben
#echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp
#MTU auf den Clients kleiner 1500 stellen
iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
#
# ENDE
```

Anzumerken ist, dass prinzipiell selbiges für einen ISDN - Anschluss gilt, jedoch ist dann im Script folgendes zu ändern: iptables -t (...) -o **ippp0** -j (...) .
Die letzten beiden Zeilen des Scriptes können weggelassen werden, wenn der Client den Dienst FTP nicht nutzen soll.
Abschließend ist es nur noch nötig, am Ende der Datei /etc/ppp/ip-up einen Verweis auf das Routing - Script zu setzen, indem wir /etc/ppp/routing ergänzen. Somit ist die Einrichtung des Routers abgeschlossen.

Nun gilt es noch den Client zu konfigurieren. Dies kann auf zweierlei Arten geschehen.

1. Über das Kommando # netconfig , das als Root ausgeführt wird. Hiernach öffnet sich eine GUI, die es ermöglicht, alle notwendigen Informationen dem Client zu liefern. Es ist darauf zu achten, dass die IP - Adresse des Servers richtig mitgegeben wird und dass der DNS - Server sowohl für Router, als auch Client gleich sein muss. Beispielsweise lautet eine DNS - Adresse für T- Online 217.5.115.7, die man mit # cat /etc/resolv.conf.saved.by.smpppd (unter SuSE 7.3 Pro) in Erfahrung bringen kann. Der Befehl „netconfig“ funktioniert problemlos bei Red Hat 7.2. Leider ist er bei nicht allen Distributionen implementiert, wobei nun Variante 2 in Betracht kommt.
2. Man kann auch manuell alle notwendigen Daten eingeben. Hierzu editieren wir die Datei /etc/resolv.conf und fügen

```
nameserver  <Adresse des Nameservers>
```

ein. Weiterhin ergänzt man beispielsweise die Datei /etc/rc.d/rc.local bzw. /etc/rc.d/boot.local um nachfolgende Einträge.

```
ifconfig  eth0  <IP-Adresse des Clients>
route  add  default  gw  <IP-Adresse des Routers>
```

Wurde die IP - Adresse des Clients schon vergeben, so kann man die erste Zeile auch weglassen.

Somit ist die Konfiguration des Netzwerkes abgeschlossen und nach einem Neustart beider Rechner steht dem Teilen der Internetverbindung nichts mehr im Wege.
Falls doch, dann sollte man zuerst versuchen, ob beide Rechner per Ping - Befehl erreichbar sind. Hierzu geben wir am Client zum Beispiel # ping -c1 <IP-Adresse des Routers> ein. Sollte der Router erreichbar sein, so ist mit # route -n zu prüfen, ob auch die Defaultroute des Clients auf den Router weißt.

Für den Fall, dass die Clients Windowsrechner sind, so ist in der Netzwerkkonfiguration als Standardgateway die IP - Adresse des Routers und in der DNS - Konfiguration die IP - Adresse des Providers (in unserem Fall 217.5.115.7) zu setzen.
Es sei erwähnt, dass dies nur für Netzwerkverbindungen und nicht für eventuell vorher eingerichtete DFÜ - Verbindungen, die nun gelöscht werden können, gilt.

Abschließend sei darauf hingewiesen, dass das Teilen einer Internetverbindung von einigen Providern nicht gestattet wird. Deshalb sollten Sie sich vorher bei Ihrem Provider diesbezüglich erkundigen.

Als Variante des bisher Beschriebenen könnte nachfolgendes Beispiel dienen, dass den Aufbau eines Netzes mit 3 Rechnern skizziert.

2. 2 So realisiere ich erfolgreich ein echtes Netzwerk unter Linux

Voraussetzungen:

3 Computer, für den Router mind. 486 DX2 100 mit mind. 32 MB RAM
Einen Hub besser Switch 10/100 Mbit, 4 Netzwerkkarten für DSL, bzw. 3 Netzwerkkarten/+ISDN oder Modem.

Definitionen:

Router:
Er bildet die Schnittstelle zwischen Welt weitem Netz und dem Intranet. Er wird daher mit 2 Netzwerkkarten ausgerüstet, eine davon wird mit dem DSL Modem verbunden, die andere verbindet unser Intranet. Seine Aufgabe ist es sämtliche Anfragen aus dem internen Netz zu Maskieren (Masquerading) und an die Gegenstelle weiter zu leiten. Sowie antworten an den ursprünglichen Absender zurück zu senden. Als Protokoll wird TCP/IP verwendet.

HUB:
Der Hub ermöglicht die Verbindung mehrerer Rechner untereinander.

Switch:
Im Gegensatz zum Hub unterteilt das Switch jede Schnittstelle in eigene Netzsegmente. Daher ist ein Switch besser gegen Angriffe von Außen speziell das Sniffing gesichert.

DNS:
Diese Abkürzung steht für Domainnameservice. Also Adressen, die sich wesentlich besser merken lassen als kryptische IP-Adressen. DNS sorgt dafür, dass eine Namensadresse in eine IP-Adresse umgesetzt werden kann. Dies ist notwendig, da die eigentliche Adressierung innerhalb eines Netzwerkes nur mittels IP-Adressen funktioniert.

Die nachfolgenden Begriffsbestimmungen habe ich von der Seite
http://www.uni-duesseldorf.de/~cappel/betriebs-kurs/ übernommen.

IP:
IP steht für Internetprotokoll Es gehört zur Netzwerkschicht. IP-Pakete können Daten verschiedener höherer Protokolle enthalten (wichtigste: TCP, UDP). IP-Pakete können über unterschiedliche Netze übertragen und "geroutet" werden. jedes Paket enthält Absender- und Zieladresse und wird unabhängig durch das Netz übertragen

TCP:
Transmission Control Protocoll Protokoll der Transportschicht.

Verbindungsorientiert:
Verbindugsaufbau erfolgt zwischen zwei "Ports" zweier Anwendungen auf zwei Endsystemen, die mittels ihrer IP-Adressen adressiert werden. Der Verbindungsaufbau erfolgt aktiv oder passiv (für Client- bzw. Server-Seite von Anwendungen). Verschiedene TCP Verbindungen einer oder mehrerer Anwendungen in einem Endsystem werden durch die Quadrupel lokale/entfernte IP-Adresse, lokale/entfernte Port-Nummer identifiziert; Port-Nummern sind 16 Bit lange Zahlen (0-65535). (Das Paar IP-Adresse/Port-Nummer wird auch Socket genannt.) Das TCP-Protokoll enthält einen gesicherten Verbindungsauf und auch Abbau (,,Three way handshake``). Die Datenübertragung innerhalb einer TCP-Verbindung ist:

Voll-Duplex: Beide Seiten können unabhängig und gleichberechtigt Daten senden.

Stream-orientiert: "Datenstrom" in beiden Richtungen: es gibt keine festen Einheiten, die genauso auf der anderen Seite ausgeliefert werden (bei ISO-OSI-Protokollen: Service Data Units).

Flusskontrolliert:
Fenstergröße, Sequenznummern gesendeter und bestätigter Daten auf Byte-(Oktett-)Ebene.

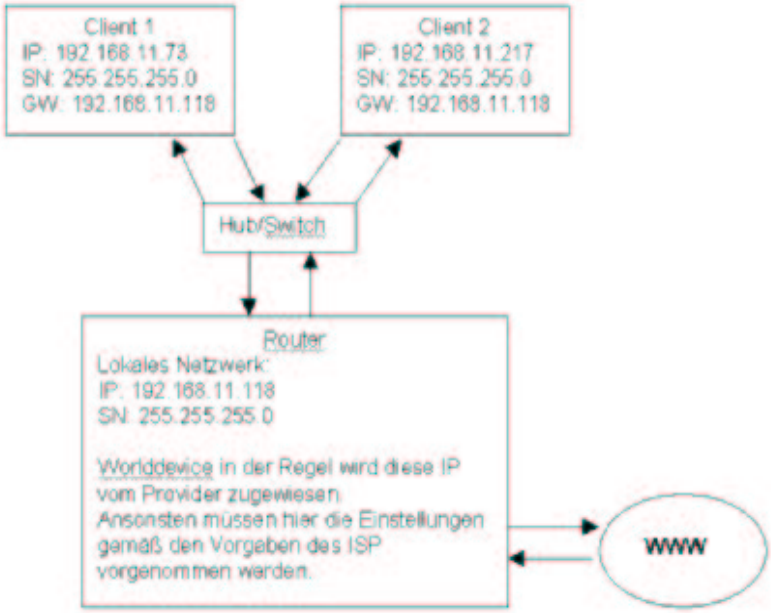
gesichert gegen Übertragungsfehler: Datensegmente können wiederholt werden, wenn sie nach Ablauf eines Retransmission-Timers nicht bestätigt wurden. Alle Abläufe in TCP sind Timer-überwacht. TCP setzt auf den Diensten von IP auf.
TCP Datensegmente werden in IP-Pakete verpackt übertragen. Spezielle Funktionen in TCP:

- Push-Funktion zur sofortigen Zustellung empfangener Daten auf der Empfangsseite.
- Übertragung von Vorrangdaten (Urgent).

Masquerading:
Für das korrekte Routing von Daten zwischen den Netzwerken ist es erforderlich, das Masquerading auf dem Router einzuschalten, damit die IP-Pakete aus dem internen Netz mit einem Header der offiziellen IP-Adresse des Worlddevice versehen werden, da IP-Adressen aus dem Privaten Bereich nicht im Internet verwendet werden können, da diese Adressen in keinem Nameserver eingetragen sind und somit Pakete nicht an entsprechende Adressen zurückgesandt werden können.

Damit sind die grundlegenden Begriffe schon einmal erklärt.

Nun wollen wir uns einmal den Aufbau unseres kleinen Netzwerks graphisch anschauen.



Reservierte IP-Adressen für lokale Netzwerke

10.0.0.0	-	10.255.255.255
127.0.0.1		Reserviert für Loopback Devices
172.16.0.0	-	172.31.255.255
192.168.0.0	-	192.168.255.255

Diese Adressen sind für die Vergabe in lokalen Netzwerken vorgesehen. Nach außen dürfen diese Adressen nicht in Erscheinung treten und müssen vom Router maskiert werden, da ansonsten die Pakete nicht beantwortet werden können.

Auflösung von Namensadressen

Für die Namensauflösung ist der so genannte Resolver zuständig. Er prüft zunächst, sofern nicht abgeschaltet, in der Datei /etc/hosts ob für die eingegebene Domaine schon ein Eintrag in Form von Domain und IP-Adresse vorhanden ist.
Ist dies nicht der Fall so wendet sich der Resolver der Datei /etc/resolv.conf zu und schaut dort, ob ein Nameservereintrag vorhanden ist.

Im nachfolgenden Beispiel ein Auszug aus beiden Dateien.

Achtung die IP-Adressen in der /etc/resolv.conf können abweichen.....!

Auszug aus der Datei /etc/hosts

```
192.168.11.73       client1.irgendeine-domain.local       client1
192.168.11.217      client2.irgendeine-domain.local       client2
192.168.11.118      gateway.irgendeine-domain.local       gateway
```

Auszug aus der Datei /etc/resolv.conf

```
search t-online.de

217.5.115.7

194.25.2.129
```

Sind alle diese Dinge erfüllt, so steht einem vernünftigen Routing nichts mehr im Wege und unser kleines Heimnetzwerk kann sich mit dem Rest der Welt verbinden.

2.3 Firewalling mit iptables

Heutzutage ist es schon fast fahrlässig ohne Firewall ins Internet zu gehen, denn man würde ja auch nicht das Haus verlassen und dabei alle Fenster und Türen offen lassen.
Einbrechen könnte man im Falle verschlossener Fenster und Türen dennoch, wobei man dem Angreifer den Einbruch erschweren würde.
Und genau dies soll unser Ziel sein. Es sei darauf hingewiesen, dass diese Firewall keine absolute Sicherheit bietet, aber sie ist für den Privatnutzer einfach zu administrieren und genügt dennoch einem hohen Maß an Sicherheit.
Bevor wir in die Materie einsteigen möchte ich noch erwähnen, dass iptables nur von Kernel - Versionen ab Version 2.4 unterstützt werden. Ferner sollte der Benutzer distributionseigene Firewalls, wie beispielsweise die SuSE Firewall 2, deaktivieren.

Im folgenden sollen die wichtigsten Befehle rund um iptables erläutert werden und uns somit zu einem sofort einsetzbaren Firewallscript führen.

Standardmäßig sind schon 3 Chains (INPUT / FORWARD / OUTPUT) gegeben und deren Policy steht anfangs auf ACCEPT.

Bezüglich einer gesamten Chain kann man:

- Die Policy für eine eingebaute Chain ändern (-P). Das geschieht beispielsweise mittels # iptables -P INPUT DROP . Es würden also hereinkommende Pakete verworfen werden (DROP). Weiterhin wäre es denkbar Pakete anzunehmen (ACCEPT) oder dies mit Rückmeldung zu verwerfen (DENY).
- Auflisten aller Regeln (- L).
- Löschen aller Regeln (-F).

Hinsichtlich einer einzelnen Regel kann man:

- Eine Regel an eine Chain anfügen (-A).
- Eine Regel in eine Chain einfügen (-I).
- Eine Regel in einer Chain ersetzen (-R).
- Eine Regel in einer Chain löschen (-D).

Hierbei ist die Reihenfolge wichtig, da der Kernel die Regeln nacheinander abarbeitet und sobald die erste Regel für ein Paket zutrifft, werden die restlichen Regeln nicht mehr beachtet.

Neben den 3 oben genannten Standardchains kann auch der Benutzer eigene Chains erstellen, wobei gilt:

- Benutzerchain definieren und benennen (-N).
- Eine (leere) Benutzerchain löschen (-X).

Abschließend gibt es noch Parameter für die jeweiligen Regeln, die da wären:

- Adresse(n) inkl. Port [Source] (-s).
- Adresse(n) inkl. Port [Destination] (-d).
- Device (-i).
- Protokoll (-p).
- Aktion [Target] (-j).

Sämtliche iptables - Variablen sind mit dem Befehl # man iptables einzusehen.
Anhand eines Beispieles sollen die bisher dargestellten Befehle verdeutlicht werden.

Wir wollen für einen IRC - Client die Ports 3334 - 3335 öffnen. Das Protokoll soll TCP sein, die Regel soll an die INPUT - Chain angefügt werden und als Device kommt ja nur ein Internetdevice (ppp0 für DSL und ippp0 für ISDN) in Frage.
Demnach muss der Befehl lauten:

```
# iptables -A INPUT -i ppp0 -p tcp --dport 3334:3335 -j ACCEPT
```

Nachdem nun die grundlegenden Funktionen von iptables erklärt wurden, wobei dies wirklich nur ein kurzer Auszug sein sollte, um den Einblick in die Thematik zu verschaffen, können wir uns an das Erstellen des Scripts heranwagen.
Bevor wir dies tun, möchte ich noch anmerken, dass eine Firewall auf zweierlei Policies basieren kann.
Einerseits ist es möglich generell alles zu blockieren und nach und nach die benötigten Dienste freizuschalten. Hierbei würden die Default - Policies auf DENY oder DROP stehen. Dies ist der sicherste Ansatz, jedoch ist er auch mit viel Wissen und Arbeit verbunden.
Andererseits kann man generell alles erlauben, was nicht ausdrücklich verboten wird. Es stehen demnach die Default - Policies auf ACCEPT.

Wir wollen uns mit dem zweiten Ansatz beschäftigen, da dieser leichter umzusetzen ist. Hierzu überprüfen wir zunächst mit # iptables -L ob die Policies auf ACCEPT stehen. Falls dem so ist, können wir nachfolgendes Script einfach so verwenden. Natürlich muss jenes noch an die Bedürfnisse des jeweiligen Benutzers angepasst werden.
In unserem Beispiel sollen die Ports für den IRC - Client (s.o.) und für ein File - Sharing - Tool geöffnet werden.

```
#!/bin/sh
#
#definierten Zustand erstellen und alle Regeln loeschen
iptables -F
iptables -t nat -F
#
#Ein Router sollte Pakete vom Typ Destination-unreachable bearbeiten
iptables -A INPUT -i ppp0 -p icmp --icmp-type destination-unreachable -j ACCEPT
#
#Edonkey-Ports oeffnen
iptables -A INPUT -i ppp0 -p tcp --dport 4661:4662 -j ACCEPT
iptables -A INPUT -i ppp0 -p udp --dport 4665 -j ACCEPT
#
#Ports fuer IRC oeffnen
#
```

```
iptables -A INPUT -i ppp0 -p tcp --dport 3334:3335 -j ACCEPT
#
#Kette erstellen, die neue Verbindungen blockt, es sei denn, sie kommen von innen
#oder es sind Antwortpakete (ESTABLISHED)
iptables -N block
iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
iptables -A block -j DROP
#
#Von Input und Forward Ketten zu dieser Kette springen
iptables -A INPUT -j block
iptables -A FORWARD -j block
#
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
#
#####ENDE########
```

Das Script kann per Link in die Datei /etc/ppp/ip-up eingebunden werden. Somit aktiviert sich die Firewall beim Starten der Internetverbindung, was auch sinnvoll ist, denn solange man offline arbeitet, benötigt man auch keine Firewall.
Durch das Festlegen der Benutzerchain block werden nur Pakete durchgelassen, die Antwortpakete (Status ESTABLISHED,RELATED) oder neue Pakete sind, die nicht über ppp0, also dem Internet, stammen. Diese Regeln werden dann noch in die INPUT - und FORWARD - Chain eingebunden. Schließlich wird das Masquerading aktiviert, das dann auf alle Pakete wirkt, welche die FORWARD - Chain erfolgreich passiert haben.

Somit reagiert der Rechner auf keinerlei Anfragen aus dem Internet und er ist demnach „unsichtbar“.
Wenn Sie weitere Dienste freischalten wollen, muss dies immer vor der Benutzerchain block erfolgen, da diese sonst nicht mehr berücksichtigt werden würden.

Ihre Firewall können Sie mittels verschiedener Webtools testen, die ich im folgenden nennen möchte:

https://grc.com/x/ne.dll?bh0bkyd2

http://www.linuxdelta.de/portscan.php

http://scan.sygatetech.com/

Abschließend sei noch einmal darauf hingewiesen, dass dies eine sehr einfache Firewall ist, deren Verhältnis zwischen Aufwand und Nutzen sehr gering ist und somit dem Anwender eine leicht konfigurierbare Lösung bietet.

Viel Spaß!

3. Tipps und Tricks

An dieser Stelle sollen Tipps und Tricks im Umgang mit dem System oder mit Utilities genannt werden, die das tägliche Leben mit Linux vereinfachen sollen, denn es ist ärgerlich, wenn man Aufgaben suboptimal löst. Schlimmer ist jedoch dann zu erfahren, dass einige viel einfacher gehen würde. Hier steht wie...

3.1 PDF - Dateien erstellen

Wer kennt nicht das Dilemma der verschiedenen Dokumentformate? Da erhält man dies und jenes und braucht zum Betrachten jedes mal das passende Programm dazu.
Vor allem aus der Windows - Welt (dies soll kein Vorwurf an die User sein) sind doch die Formate aus dem ganzen Officepaketen für den Linuxuser meist nicht korrekt lesbar. Zwar gibt es für Linux Programme wie Star Office, die versuchen eine Kompatibilität zu Microsoft Officepaketen herzustellen, aber in Redmond wird in der Zwischenzeit an einer neuen Version getüftelt, die nunmehr inkompatibel ist. Als Folge bekommen wir Dokumente, die zwar lesbar sind, aber jegliche Formatierungen sind dahin.
Um dieses Dilemma zu lösen gibt es zwei einfache Lösungen. Einerseits kann man die Dateien im HTML- oder Text - Format speichern und so dem Gegenüber zugänglich machen, wobei der Nachteil darin liegt, dass auch hierbei wiederum die Formatierungen verloren gehen, jedoch ist das Dokument lesbarer.
Andererseits kann man das Problem mittels dem plattformübergreifenden PDF - Format lösen. Hierbei bleiben alle Formatierungen erhalten, der Gegenüber kann das Dokument lesen und drucken, wie es ursprünglich erstellt wurde. Ferner gibt es mehrere kostenlose PDF - Reader zum Download, wobei der Acrobat Reader als Standard zu nennen ist.
So nun genug der Einleitung....

Das Erstellen von PDF - Dokumenten geschieht in 2 Schritten.

1. Wir wählen in der jeweiligen Anwendung den Menüpunkt „Datei drucken“ und anschließend die Option „Ausgabe in Datei“. Nun gilt es der Datei einen Namen zu geben und im richtigen Format zu speichern. Da Linux eh mit Postscript - Druckertreiber arbeitet, gibt man also folgendes ein:
   <Dateiname>.ps
   Nun ist nur noch die Datei zu „drucken“....
2. Abschließend wechselt man in das Verzeichnis, in dem die gerade erstellte Postscriptdatei liegt und führt den Befehl
   # ps2pdf12 <Dateiname>.ps <Dateiname>.pdf
   aus. Sollte dies nicht auf Anhieb funktionieren, so ist das Paket „ps2pdf12“ nach zu installieren. Die Postscriptdatei kann nun getrost gelöscht werden.

Somit kann jeder innerhalb weniger Sekunden eine PDF - Datei erzeugen und diese jedem problemlos zugänglich machen.

3.2 Benutzerrechte editieren

Vielfach stellt sich die Frage für einen Linuxeinsteiger wo denn die ausführbaren Dateien unter Linux zu finden sind.
Also wollen wir doch mal schauen, wie wir herausfinden, ob eine Datei ausführbar ist.

Mit # ls -l kann man sich die Rechte und Besitzer einer Datei anzeigen lassen. Das könnte in etwa so aussehen:

```
-rwxr-x---   1 root     root        42324 Jul 29  2000 route
```

In diesem Falle sehen wir, dass der Befehl route der Gruppe root und dem Benutzer root zugeordnet wurde. Damit ist diese Datei für andere Gruppen und Benutzer nicht ausführbar.

Nun stehen aber ganz vorne seltsame Abkürzungen, die wir uns nun einmal näher anschauen wollen.

Der erste Bindestrich bedeutet, dass es sich hierbei um eine Datei handelt. Ein d an erster Stelle steht für directory, ein l für Link.

Es folgen die Rechte.

r für read
w für write
x für execute (Im Falle eines Verzeichnis gibt diese Option an, ob in das
Verzeichnis gewechselt werden darf)

Wie aber ändert man denn nun die Rechte an einer Datei?

Dafür gibt es unter Linux ein Kommando namens chmod.

Nehmen wir mal an, es befindet sich eine Datei test in unserem Homeverzeichnis, welche die Funktion hat, das aktuelle Datum auszugeben. Der Inhalt der Datei sieht dann folgendermaßen aus:

```
#!/bin/bash
date
```

[Anmerkung: mit # touch test wird die Datei erstellt und der Inhalt kann mittels eines beliebigen Editors bearbeitet werden.]

Mit # ls -l zeigen wir uns nun die Rechte an.
Das sieht nun so aus:

```
-rw-r--r--   1 tux users        17 Mai  4 12:54 test
```

Wenn wir nun mit # ./test versuchen die Datei auszuführen wird die Enttäuschung groß sein, weil wir feststellen werden, dass es gar nicht geht.

```
bash: ./test: Keine Berechtigung   --> so wird die Fehlermeldung aussehen
```

Aber wir kennen ja jetzt unseren chmod Befehl und können Abhilfe schaffen.

Also einfach mal # chmod 750 test eingeben und nun sieht das ganze schon anders aus.
[Anmerkung: r = 4 ; w = 2; x = 1]

```
-rwxr-x---   1 tux users        17 Mai  4 12:54 test
```

Wenn wir nun # ./test aufrufen erhalten wir das folgende Ergebnis:

Sam Mai 4 12:59:28 MEST 2002

Hat doch prima geklappt, oder? Und so schwer war es doch gar nicht. Also probiert doch einfach mal aus, denn schief gehen kann dabei nichts, so lange ihr die Testdatei hernehmt.

Näheres zum chmod Befehl könnt Ihr in der Manual von chmod nachlesen.

```
# man chmod
```

Viel Spaß beim tüfteln und basteln.
Eure Fragen beantworten wir gerne im Forum oder im IRC-Chat.
In den IRC-Chat gelangt Ihr z.B. über das Chatapplet auf unserer Chatseite.

3.3 Linken - aber richtig !

„Hmpf....! Schon wieder das Verzeichnis wechseln und dabei habe ich doch schon tausend Fenster offen“.

Du standest sicher schon einmal vor einem ähnlichen Problem.
Da hast Du Dir auf Deinem Webserver eine niegelnagelneue Webseite erstellt und ständig musst Du daran herum basteln. Und weil das ja nicht genug ist geht es los Terminalfenster auf und

```
# cd /var/webserver../...
```

Das nervt natürlich auf die Dauer, aber auch hier gibt es Abhilfe.
Wie wäre es, wenn man mit `# cd webserver` einfach in dem Verzeichnis landen würde, in dem sich die HTML-Dateien befinden?

Nichts leichter als das:

```
# ln -s /var/webserver  /home/tux/webserver
```

Allgemein wird der Link also gesetzt mit:

```
# ln -s <Quelle> <Ziel>
```

Mehr Informationen zum Linken findest Du im übrigen in der Manual von ln.

Viel Erfolg beim Ausprobieren.

3.4 Manuals - Oder wie helfe ich mir selbst?

Der erste Schritt, bei Problemen mit einem Programm oder einem Tool sollte immer das Lesen der Dokumentation sein. Unter Linux findet man häufig so genannte Kurzreferenzen, eine Art Liste, in der man nachschauen kann mit welchen Optionen ein Programm gestartet werden kann.
Hierbei handelt es sich um das sogenannte Manualsystem.

Mit # man [Programmname] kann solch eine Kurzreferenz aufgeschlagen werden.

Schauen wir uns doch mal die Manual von ls an.
Wir finden zunächst einige Oberkategorien die meist in der Reihenfolge

Bezeichnung
ls - (list) zeigt den Inhalt eines Verzeichnisses
Hier wird die Wirkung des Befehls ls erklärt.

Syntax
ls [-abcdgiklmnpqrstuxABCFLNQRSUX1]
Hier werden die Optionen mit denen das Programm gestartet werden kann angezeigt

Beschreibung

ls gibt den Inhalt der Verzeichnisse des Dateisystems an.

Das Standardausgabeformat von ls hängt vom Typ des Aus
gabegerätes ab. Auf einem Terminal ist die mehrspaltige
Ausgabe das Standardformat. In allen anderen Fällen wird
die Ausgabe einspaltig ausgeführt.

Das Verhalten des ls Kommandos läßt sich nicht mehr durch
Umbenennen in ll dir vdir etc. verändern. Stattdessen
sind die Kommandos dir und vdir als separate Binärdateien
mit entsprechenden Standardformaten verfügbar.

Wie wir hier sehen wird das Programm in dieser Kategorie näher beschrieben.

Optionen

-a zeigt alle Dateien im Verzeichnis, auch die deren Name mit . beginnt

Würde man nun ls -a auf der Konsole eingeben so würde dies dazu führen, dass der gesamte Verzeichnisinhalt angezeigt wird inklusive der versteckten Dateien und Verzeichnisse (a steht im übrigen für all also alles).

Versuch doch einfach mal herauszufinden mit welcher Option Du Dir die Rechte einer Datei anschauen kannst.

Viel Erfolg

3.5 X - Window - Server - Session freigeben

Wer hat nicht schon einmal versucht eine Anwendung zu starten, die Root - Privilegien benötigt, obwohl man typischerweise an einem Linux - System als User angemeldet ist?
Man versucht sich auf der Konsole mittels

```
# su
```

als Super-User anzumelden, versucht die Anwendung, die eine X - Session benötigt, auszuführen und bekommt die Fehlermeldung „...refused by server“.
Der Grund hierfür ist, dass Linux ein Multi - User - Betriebssystem darstellt und die Fehlermeldung auf eine implementierte Sicherheitsfunktion zurückzuführen ist, denn es kann kein anderer User auf eine laufende X - Session zugreifen - **und das gilt auch für den so allmächtigen Root!**

Die Frage ist nun, wie man das Problem umgeht. Man könnte sich beispielsweise ausloggen und als Root wieder anmelden, aber das wäre ja viel zu umständlich.

Abhilfe schafft ein kleiner Befehl. Bevor man in der Konsole sich als Root anmeldet, gibt man als User den Befehl

```
# xhost +
```

ein.
Nun kann man sich als Root anmelden und die benötigte Anwendung innerhalb der laufenden

X - Session ausführen.

Achtung ! Während dieser Phase kann jeder User innerhalb des lokalen Netzes auf Ihre X - Session zugreifen. Deshalb ist diese Vorgehensweise nur für Einzelplatzsysteme ratsam.

Hat man seine Arbeit als Root erledigt, loggt man sich mittels [STRG] + [D] wieder aus und führt als User den Befehl

```
# xhost -
```

aus, um die Freigabe der X - Session wieder aufzuheben.

Viel Spaß!

4 .Workshop s

Ziel dieses Kapitel ist es, dem User spielend den Umgang mit dem zu ermöglichen, wobei aktive Mitarbeit nun gefragt ist.
Für eventuelle Schäden am System, die durch einen Workshop hervorgerufen wurden, übernehmen wir natürlich keine Haftung.... :-)

4.1 Dateien und Verzeichnisse

Dieser Workshop hat die Aufgabe, dem Linux - Einsteiger einen Einblick in die Arbeit mit der Shell und deren funktionsweise zu ermöglichen. Natürlich besitzt die Shell auch noch weitere Funktionen als die Arbeit mit Dateien und Verzeichnissen, aber diese sind meiner Ansicht nach für einen Einsteiger erst einmal nicht elementar.
Es sei darauf hingewiesen, dass es keinen Sinn macht, diesen Workshop einfach nur zu lesen, sondern es ist hier wirklich aktive Mitarbeit angesagt.

4.1.1 Teil 1

Bevor wir einsteigen, halte ich es für sinnvoll einen weiteren Benutzer anzulegen, damit man problemlos Dateien und Verzeichnisse anlegen, bearbeiten und löschen kann, ohne etwas am laufenden System zu

gefährden.
Hierzu loggen wir uns als root mit
su
ein. Und geben nacheinander folgende Befehle ein.

```
# groupadd -g 9000 workshop
# useradd -u 9001 -d /home/workshop workshop
# usermod -g workshop -G workshop -s /bin/bash workshop
# mkdir -p /home/workshop
# chown -R workshop /home/workshop
# passwd workshop
```

Nun wechseln wir mit # su workshop den User und mit # cd /home/workshop in das entsprechende Homeverzeichnis.

Einen Eintrag in der Datei .bashrc mit dem Inhalt
export PS1="[\W]> "
erstellen wir mit
echo 'export PS1="[\W]> "' >/home/workshop/.bashrc

Nun können wir mit dem eigentlichen Teil des Workshops beginnen.
Da uns der Shell-Prompt keine sehr genau Aussage liefert, in welchem Verzeichnis wir uns denn nun befinden, können wir uns dem Kommando (print working directory)
pwd
bedienen. Als Output werden wir nun
/home/workshop
erhalten. Also befinden wir uns in dem eben dargestellten Verzeichnis.
Nun möchten wir aber auch wissen, was sich in diesem Verzeichnis befindet und welche Unterverzeichnisse es gibt.
ls -al
heißt das Zauberwort. Die Parameter -a und -l können natürlich auch weggelassen werden, jedoch ermöglichen diese einen tieferen Einblick, denn -a zeigt auch die versteckten Dateien und Verzeichnisse an, welche mit einem Punkt beginnen und der Parameter -l erweitert die Ansicht, d.h. es werden uns mehr Informationen geliefert. Beispielsweise liefert uns jener Befehl folgendes:

```
drwx------  4 workshop root          4096 May  6 14:37 .
drwxr-xr-x  6 root     root          4096 May  6 13:59 ..
-rw-------  1 workshop workshop   484 May  6 15:10 .bash_history
-rw-r--r--  1 workshop workshop    20 May  6 14:39 .bashrc
```

Die hier angegebenen Spalten haben von links nach rechts folgende Bedeutung:
d=Verzeichnis, Zugriffsrechte (lesen, schreiben, ausführen), Anzahl der Verweise, Benutzer, Gruppe, Größe, Datum der letzten Änderung, Dateiname. Mit den Zugriffsrechten wollen wir uns später befassen.
Was uns nun auffällt ist, dass sich im Verzeichnis /home/workshop nur versteckte Dateien und Verzeichnisse befinden, also würde uns der Befehl # ls nichts anzeigen, da wir es mit einem leeren Verzeichnis zu tun haben.

Da ein leeres Verzeichnis eher für ausgesprochene Langweile sorgt, werden wir nun ein paar Dateien und Verzeichnisse erstellen.
Befassen wir uns zunächst mit dem Erstellen von Dateien. Eine leere Datei kann ganz einfach mit dem Befehl # touch <dateiname> erstellt werden, also kreieren wir mittels
touch linux.txt; touch tux.txt; touch torvald.txt
jene 3 Dateien, wobei das Semikolon in diesem Befehl dazu führt, dass die Befehle nacheinander ausgeführt werden, falls der vorangehende Befehl fehlerfrei ausgeführt wurde.

Im nächsten Schritt wollen wir auch noch 2 Unterverzeichnisse anlegen. Also geben wir hierfür
mkdir texte; mkdir programme
ein. Nun prüfen wir mit # ls -l , was sich jetzt in unserem Homeverzeichnis befindet und siehe da:

```
 -rw-r--r--  1 workshop workshop        0 May  6 16:17 linux.txt
drwxr-xr-x  2 workshop workshop     4096 May  6 16:31 programme
drwxr-xr-x  2 workshop workshop     4096 May  6 16:31 texte
 -rw-r--r--  1 workshop workshop        0 May  6 16:17 torvald.txt
 -rw-r--r--  1 workshop workshop        0 May  6 16:17 tux.txt
```

Folglich haben wir 3 leere Dateien und 2 leere Verzeichnisse, aber das ist doch schon einmal ein Anfang.

Jetzt ist es unser Anliegen, eine Textdatei namens cool.txt mit dem Inhalt "Linux ist cool" im Verzeichnis /home/workshop/texte zu erzeugen, die Frage ist nur, wie wir da vorgehen sollen...
Es kann schon einmal nicht schaden, wenn man mit # cd texte (wenn man sich im Homeverzeichnis befindet) oder mit # cd /home/workshop/texte in das jeweilige Verzeichnis wechselt. Mit # pwd können wir

uns das aktuelle Verzeichnis, in dem wir uns befinden, wiederum anzeigen lassen.[1]

Nunmehr wollen wir die Textdatei erstellen. Dies geschieht beispielsweise durch Benutzung des mächtigen Editors "*vi*", der v.a. bei Systemadministratoren beliebt ist. Also folgt als Befehl:

```
# vi cool.text
```

Der Texteditor ist jetzt gestartet und man muss ihm nur noch mitteilen, dass man etwas eingeben (engl. insert) möchte. Hierfür drücken wir die Taste "I" (am unteren Fensterrand erscheint insert) und geben den Text "Linux ist cool ein". Durch Drücken der ESC-Taste erfährt der Editor, dass wir mit der Eingabe fertig sind und gespeichert wird die Datei durch die Eingabe von ":wq" (ohne Anführungszeichen).

Mehr möchten wir an dieser Stelle nicht auf den Editor *vi* eingehen.

`# ls` zeigt uns nun an, dass die Datei cool.txt vorhanden ist.

Jetzt wäre es natürlich ganz fein, wenn man sich den Inhalt der Datei auch gleich ansehen könnte. Und wer hätte es gedacht? Auch hierfür hat Linux ein passendes Kommando.

```
# cat cool.txt
```

liefert uns den Text " Linux ist cool ".

Was wäre, wenn der Inhalt jener Datei sehr lang wäre? Dann würde nämlich der Befehl "cat" sehr unbequem sein, denn der gesamte Output würde auf einmal dargestellt werden. Hier schafft der Befehl "less" Abhilfe, der den Output seitenweise präsentieren würde. Auf die nächste Seite käme man durch Drücken der Spacetaste und beenden könnte man die Prozedur durch die Eingabe von "q".

Der Befehl lautet also

```
# less cool.txt
```

Nach erledigter Arbeit würden wir wieder in unser Homeverzeichnis[2] wechseln wollen, aber müssen wir da wirklich

```
# cd /home/workshop
```

eingeben? Geht das nicht einfacher....?

Und auch hier hat Linux wieder eine Lösung parat, nämlich den Befehl

```
# cd ..
```

Dieser bewirkt, dass wir uns in das nächst höhere Verzeichnis bewegen.

1Anmerkung: Die Eingabe von langen Datei- oder Verzeichnisnamen kann durch das Drücken der Tabulatortaste vekürzt werden. Beispielsweise liefert work[Tab] den vollen Ordnernamen workshop.

2 Um in das Homeverzeichnis zu wechseln könnte man auch

```
# cd ~
```

schreiben.

Zum Schluss des ersten Teils der Workshops "Dateien und Verzeichnisse" wollen wir die Datei cool.txt aus /home/workshop/texte nach /home/workshop kopieren und gleichzeitig den Namen der Kopie in copycool.txt ändern.
Nichts leichter als das, denn auch hierfür gibt es einen einfachen Befehl:

```
# cp /home/workshop/texte/cool.txt /home/workshop/copycool.txt
```

wobei links die Quelldatei und rechts die Zieldatei inklusive deren absolute Verzeichnisse stehen.

Viel Spaß!

4.1.2 Teil 2

Bevor es nun weiter geht, stellen wir sicher, dass wir uns als User "Workshop" in dessen Homeverzeichnis befinden. Wie dies funktioniert, möchte ich an dieser Stelle nicht noch einmal erläutern. Sollte jemand diesbezüglich dennoch Probleme haben, so ist es ratsam, den ersten Teil des Workshops zu wiederholen.

Zum Aufwärmen wollen wir uns einer sehr einfachen Aufgabe widmen, nämlich dem Umbenennen und Verschieben von Dateien.
Könnten beide Vorgänge mit ein und demselben Befehl bewerkstelligt werden?
Die Antwort lautet "Ja", denn prinzipiell ist der Ablauf der selbe. Die Quelldatei wird gelöscht und eine neue Zieldatei mit dem Inhalt der Quelldatei wird angelegt - und wo diese Datei angelegt wird bleibt uns überlassen, was eben gleichbedeutend mit dem Verschieben von Dateien ist.
Wollen wir uns diese beiden Möglichkeiten einmal näher ansehen:

1. Die Datei copycool.txt im Homeverzeichnis soll in die Datei supercool.txt umbenannt werden.
   Der Befehl lautet hierfür einfach:
   ```
   # mv copycool.txt supercool.txt
   ```
   mit `# ls` können wir unser Ergebnis betrachten.
2. Nun soll die Datei supercool.txt in das Verzeichnis /home/workshop/texte verschoben werden.
   Auch dies ist äußerst einfach und wird mit folgenden modifizierten Befehl geleistet:
   ```
   # mv supercool.txt /home/workshop/texte
   ```
   mit `# ls` im Homeverzeichnis sehen wir, dass die Datei nicht mehr vorhanden ist, und mit `# cd ~/texte ; ls` erkennen wir, dass die Datei sich nun im Verzeichnis "texte" befindet.

Nachdem wir nun wissen, wie man Dateien erstellt, kopiert, umbenennt und verschiebt, wäre es natürlich auch hilfreich zu wissen, wie man Dateien wieder löscht.

Und auch dies wir durch einen simplen Befehl durchgeführt, den wir im Folgenden betrachten möchten. Hierzu wechseln wir mit # cd ~ ins Homeverzeichnis und lassen uns mit # ls -l den Inhalt anzeigen, der folgendermaßen aussehen müsste:

```
-rw-r--r--    1 workshop workshop        0 Mai  6 16:17 linux.txt
drwxr-xr-x    2 workshop workshop     4096 Mai  6 16:31 programme
drwxr-xr-x    2 workshop workshop     4096 Mai 16 13:39 texte
-rw-r--r--    1 workshop workshop        0 Mai  6 16:17 torvald.txt
-rw-r--r--    1 workshop workshop        0 Mai  6 16:17 tux.txt
```

Unser Opfer sei nun die Datei "linux.txt". Diese löschen wir mit

rm linux.txt

und überprüfen die Aktion wiederum mit # ls und siehe da, die Datei hat sich ins Nirvana verabschiedet.

Nun haben wir schon wichtige Grundlagen im Umgang mit Dateien erlangt, aber können wir unsere Kenntnisse auch auf Verzeichnisse anwenden? Prinzipiell würde ja nichts dagegen sprechen, oder?
Der geneigte Leser erkennt natürlich gleich, worauf wir hinaus wollen, denn 2 rhetorische Fragen hintereinander sind doch "... wie der Schlag mit dem Zaunpfahl".
Also versuchen wir folgendes im Homeverzeichnis:

rm texte

und wir erhalten als Antwort - wer hätte es auch anders gedacht ? - rm: »texte« ist ein Verzeichnis als Fehlermeldung.
Wir halten also fest, dass sich Verzeichnisse also nicht so einfach löschen lassen als einzelne Dateien - und dies ist auch gut so, denn das Löschen von Verzeichnissen kann natürlich schwerwiegende Folgen haben, aber hierzu später mehr.
Als gültiger Befehl zum Löschen des Verzeichnisses käme der Befehl

rmdir texte

in Betracht, aber siehe da: schon wieder eine Fehlermeldung... rmdir: »texte«: Das Verzeichnis ist nicht leer
Wir halten also weiterhin fest, dass sich nicht - leere Verzeichnisse noch schwieriger löschen lassen.
Lassen wir uns aber nicht entmutigen und versuchen den Befehl am leeren Verzeichnis /home/workshop/programme mittels

rmdir /home/workshop/programme

und was ist passiert?
Das Verzeichnis ist gelöscht!
Aber was machen wir nun, wenn wir ein nicht - leeres Verzeichnis löschen wollen? Müssen wir da alle Dateien und Unterverzeichnisse rekursiv löschen?

Natürlich nicht, denn auch hierfür gibt es wieder einen einfachen Befehl, aber wir wollen uns kurz einem anderen Thema widmen ehe wir ein nicht - leeres Verzeichnis löschen.

Wie wir schon wissen, erstellen wir mit Hilfe des Befehls
mkdir <Verzeichnisname>
ein neues Verzeichnis. Nehmen wir einfach einmal an, dass wir ein Ober - und ein Unterverzeichnis erstellen wollen. Was fällt auf?
Ganz genau. Man müsste mit obigen Befehl das Oberverzeichnis erstellen, dann in dieses Verzeichnis wechseln und dann das Unterverzeichnis erstellen. Und das ist schon wieder viel zu umständlich!
Abhilfe schafft hierbei ebenfalls ein kleiner Befehl, der uns die Arbeit abnimmt, wenn wir beispielsweise die Verzeichnisse /home/workshop/ober/ und /home/workshop/ober/unter/ anlegen wollen, nämlich:
mkdirhier /home/workshop/ober/unter .
Einfach, oder?

Abschließend wollen wir nun endlich ein nicht - leeres Verzeichnis löschen. Da wir das Verzeichnis "texte" zu einem späteren Zeitpunkt noch benötigen, löschen wir das gerade erstellte nicht - leere Verzeichnis /home/workshop/ober/ . Dieses ist nicht - leer, weil sich ja in diesem das Verzeichnis "unter" noch befindet.
Das Zauberwort zum Löschen nicht - leerer Verzeichnisse lautet:
rm -r /home/workshop/ober/
Und nun sind alle Verzeichnisse und Dateien vom Verzeichnis /home/workshop/ober/ abwärts verschwunden.

Jetzt dürfte der Leser die Gefahr dieses Befehls erkennen, denn ein
rm -rf /
als **Root** ausgeführt, würde dazu führen, dass das ganze Dateisystem - also **ALLES !!!** - gelöscht werden würde.

Folglich ist das rekursive Löschen von Verzeichnissen wirklich mit Vorsicht zu genießen.... Wir wissen jetzt warum!
Weiterhin viel Spaß!

4.1.3 Teil 3

Nun befinden wir uns schon fast am Ende dieses kleinen, einführenden Workshops. Bevor wir aber den Benutzer "Workshop" wieder löschen, da wir diesen dann nicht mehr für diesen Zweck benötigen, wollen wir uns noch ein wenig dem Thema "Dateien und Verzeichnisse" widmen.

Wir wechseln nun in das Verzeichnis /home/workshop/texte und finden dort 2 Textdateien vor, nämlich "cool.txt" und "supercool.txt". Wenn wir uns noch an den ersten Teil des Workshops erinnern, so ist die Datei "supercool.txt" eine Kopie der erstgenannten Datei.
Wie wir wissen, können wir uns den Inhalt einer Datei entweder mittels
cat <Dateiname> oder # less <Dateiname>
anzeigen lassen.
Wenn wir also die beiden Dateien auf ihre Gleichheit überprüfen wollten, müssten wir einen der beiden Befehle für beide Dateien ausführen und deren Output vergleichen. Dies ist bei den gerade verwendeten Dateien noch sehr einfach, weil deren Inhalt einfach nur aus dem kurzen Satz
Linux ist cool
besteht. Schwieriger wäre es natürlich, wenn die jeweiligen Dateien umfangreicher wären.
Jedoch stellt der geneigte Linux - Anwender fest, dass dieses Vorgehen doch schon wieder viel zu umständlich wäre und er erwartet, dass Linux uns auch hierfür einen passenden Befehl bietet.... Und zu Recht!
Der Befehl lautet # cmp (engl. compare) .
Also würden wir, wenn wir beide Dateien vergleichen wollten,
cmp cool.txt supercool.txt
eingeben... Und was passiert ? - Nichts, denn der Befehl liefert nur dann ein Feedback, wenn die Dateien eben nicht identisch sind.
Um dies zu verdeutlichen legen wir uns mittels
vi linux.txt
eine neue Datei mit dem Inhalt : "Linux macht Spaß!" an.
(Der Umgang mit dem Texteditor vi wurde im ersten Teil des Workshops kurz beschrieben. Es kann auch ein anderer Editor verwendet werden.)
So, nun vergleichen wir die gerade erstellte Datei mit einer der oberen Dateien.
cpm linux.txt cool.txt
liefert als Rückmeldung
cool.txt linux.txt differ: char 7, line 1 ,
was bedeutet, dass sich beide Dateien ab dem 7. Zeichen in der ersten Zeile unterschieden und dies leuchtet auch ein, da am Anfang beider Dateien "Linux" steht und dieses Wort 6 Zeichen aufweißt.

Somit haben wir einen sehr einfachen Befehl gesehen, der es uns ermöglicht, Dateien ohne großen Aufwand auf Identität zu prüfen.

Abschließend wollen wir uns noch kurz der "Systemverwaltung" zuwenden. Natürlich ist dies ein breit gefächertes Thema, aber wir wollen uns dem Ganzen nur aus der Sicht des Users widmen - und nicht aus der Sicht des Systemadministrators.

Was uns natürlich als User interessiert, ist der benötigte Speicherplatz einer Datei oder eines Verzeichnisses auf der Festplatte (engl. disk usage).
Beispielsweise liefert
du -h cool.txt
als Output
4.0k cool.txt
oder
du -h /home/workshop/texte/
zeigt die Größe des Verzeichnisses, nämlich
16k /home/workshop/texte .

Viel mehr muss man zu diesem Kommando eigentlich nicht mehr sagen, da dies eigentlich selbsterklärend ist.
Was uns zudem noch interessieren könnte, wäre der komplett belegte Speicherplatz auf der Festplatte, den wir auch sehr einfach über ein einziges Kommando in Erfahrung bringen können, nämlich:
df -h .
Wir erhalten beispielsweise als Antwort:

```
Dateisystem          Größe Benut  Verf Ben% montiert auf
/dev/hda7             18G  4.5G   13G   26%   /
/dev/hda5             15M  6.4M  7.9M  45%   /boot
shmfs                 62M     0    61M   0%  /dev/shm
```

Auch diese Meldung ist selbsterklärend... Wir erkennen, dass wir noch genügend Speicher, nämlich rund 13 Gigabyte, frei haben.

Was wäre aber, wenn dem nicht so wäre, und wir uns langsam - aber sicher - dem Ende unseres Festplattenspeichers nähern würden?

"**Backup**" ist die vernünftigste Antwort, denn löschen wollen wir doch alle nicht, oder?
Die Frage ist nur, wie wir das am einfachsten bzw. am schnellstem bewerkstelligen können, denn jede Datei einzeln auf CD oder andere Wechselmedien zu pressen, wäre Sisyphusarbeit.
Des Rätsels Lösung wäre die Benutzung eines Archivprogrammes, mittels dessen es beispielsweise möglich wäre, ganze Verzeichnisse und der darin enthaltenen Dateien und Unterverzeichnisse zu "packen" - und natürlich ist auch dies mit Linux realisierbar.
Wir wollen deshalb zum Schluss des Verzeichnis /home/workshop/texte archivieren und tippen:

```
# tar cfvz backup.tar.gz texte/ .
```

Somit erhalten wir die Datei "backup.tar.gz", die nun alle Dateien ab dem Verzeichnis "texte" enthält.
Das Backup wollen wir nun auch einmal kurz testen und löschen daher das gesicherte Verzeichnis mit:

```
# rm -rf /home/workshop/texte/
```

und spielen anschließend vom Homeverzeichnis aus das Backup wieder zurück, indem wir folgendes eingeben:

```
# tar xfvz backup.tar.gz .
```

Den Inhalt des gerade erstellten Backups hätten wir uns auch mit

```
# tar  tar tvzf backup.tar.gz
```

anzeigen lassen können, was dann nachfolgendes Aussehen hätte.

```
drwxr-xr-x workshop/workshop   0 2002-05-23 15:46:23 texte/
-rw-r--r--    workshop/workshop 15 2002-05-06 17:05:25 texte/cool.txt
-rw-r--r--    workshop/workshop 18 2002-05-23 13:52:40 texte/linux.txt
-rw-r--r--    workshop/workshop 15 2002-05-23 13:47:51 texte/supercool.txt
```

Zum Schluss möchte wir natürlich noch den User "Workshop" löschen, da wir diesen für unsere Zwecke nicht mehr benötigen.
(Natürlich kann man auch jenen User zum Experimentieren auf dem System belassen.)
Dies geschieht ganz einfach durch die Eingabe von

```
# userdel -r workshop
```

als Root, wobei darauf hingewiesen wird, dass dies nur funktioniert, falls alle Prozesse des Benutzers "Workshop" beendet wurden.
Ich möchte noch anmerken, dass alle hier verwendeten Befehle (und natürlich noch viel mehr...) ausgiebig dokumentiert sind, und deshalb können alle notwendigen Parameter über

```
# man <Befehl>
```

in Erfahrung gebracht werden.
Viel Erfolg!

5. Sonstiges

Linux hat natürlich auch noch seine Stärken in andern Bereichen und das wollen wir nun in diesem Kapitel aufzeigen.

5.1 Edonkey - Installationsanleitung

In den letzten Jahren hat sich Filesharing zu einer immer beliebteren Disziplin entwickelt und das Programm Edonkey (www.edonkey2000.com) ist meiner Meinung nach hierfür eines der besten Tools, da es von mehreren Quellen gleichzeitig downloaden kann, ein gewisser Uploadzwang besteht und Resuming unterstützt wird, was natürlich den Transfer größerer Datenmengen begünstigt.

Die Installationsanleitung bezieht sich auf den offiziellen CORE und der offiziellen GUI, die über http://users.aber.ac.uk/tpm01/guihome.html bezogen werden kann.

Wir möchten jedoch anmerken, dass möglicherweise Inhalte von anderen Usern angeboten werden, die nicht legal sind, deshalb distanzieren wir uns ausdrücklich von den angeboteten Inhalten und weisen die Nutzer darauf hin, dass sie diesbezüglich selbstverantwortlich sind.

Nun aber zur eigentlichen Installation:

- Nachdem wir sowohl die GUI als auch den CORE im Verzeichnis /home/dein_username gespeichert haben, wechseln wir mit # cd /home/dein_username in jenes Verzeichnis.
- Nun entpacken wir den CORE mit dem Befehl # gunzip donkey_s_59-3.gz und setzten die entsprechenden Rechte, damit der CORE auch vom Benutzer gestartet werden kann. Dies geschieht durch die Eingabe von # chmod 755 donkey_s_59-3 .
- Jetzt kann der CORE gestartet werden, indem wir ihn mit # ./donkey_s_59-3 aufrufen.
- Durch die Eingabe von pass <benutzername> <passwort> setzen wir einen beliebigen Benutzernamen und ein Passwort, überprüfen dies mittels der Eingabe von vo und beenden den CORE über den Befehl q . Somit ist die Konfiguration des COREs erledigt, der Rest kann über die graphische Oberfläche erfolgen.
- Die GUI starten wir nun über den Befehl # /home/dein_username/linux_gui_alpha_unstable .
- Nun wird der „connect to“ - Dialog erscheinen und dieser wird die Fehlermeldung bringen, dass der local CORE nicht läuft. Deshalb klicken wir auf „spawn local core“. Sollte dies nicht funktionieren, so ist es nötig, dass der Pfad zum CORE, also /home/dein_username/donkey_s_59-3 korrekt eingegeben wird und

noch einmal „spawn local core“ ausgewählt wird.
Sollte auch dies fehlschlagen, so kann der CORE auch manuell über die Konsole gestartet werden (s.o.).

Nun sollte der Client über die GUI problemlos laufen.

Abschließend möchte ich noch ein paar Anmerkungen zu den Einstellungen machen, damit der Esel auch ordentlich seinen Dienst erledigt. Diese sind nachfolgender Tabelle zu entnehmen und gelten für T - DSL - Anschlüsse.

maximum download speed	96
maximum upload speed	13
max. connections	500
server rotate	0
always stay connected	Ja
automatically remove dead servers	Ja
shutdown core on exit	Ja
autospawn local core on startup if needed	Ja
try to restart core on crash	Ja
automatically fetch new servers ...	Ja

Diese Einstellungen sind natürlich unverbindlich, denn der Esel ist ein etwas widerspenstiges Filesharing - Tool, bei dem der Benutzer mit etwas Experimentieren gefragt ist, jedoch sollten hiermit ganz gute Ergebnisse erzielt werden.
Viel Spaß!

6. Autorenverzeichnis

Hiermit danken wir allen Autoren, die Ihre Arbeitskraft und Freizeit diesem Projekt zur Verfügung gestellt haben.
Anhand der Indizies aus dem Inhaltsverzeichnis lassen sich die Autoren der jeweiligen Artikel ableiten.
Beteiligt waren:

1) Karsten Nordsiek
2) Atze
3) Steffen Nordmann