Mit LINUX ins Internet

Hubert Pitner

LINUX - das Betriebssystem aus dem Internet - lässt sich natürlich von jedem beliebigen Telefonanschluss per Modem mit dem Internet verbinden. Der Haken daran ist nur, dass jede LINUX Distribution ihre eigenen Werkzeuge zur "Vereinfachung" des Anschlusses an das Internet mitbringt. "kppp" ist z.B. ein solches Tool, welches unter der Bedienoberfläche "KDE" läuft und ein echtes Windows-98-Feeling bietet.

Diese kurze Anleitung sollte es jedem Anwender ermöglichen, seinen LINUX-Rechner mit dem Internet zu verbinden.

Was wird für den Internet Anschluss benötigt?

a ein problemlos laufender LINUX Rechner.

b ein Modem (je schneller desto besser) als Verbindung zwischen dem Rechner und dem Telefonnetz.

c einen User- und einen Mailbox Account bei einem beliebigen Provider.

d auf dem LINUX Rechner müssen Sie Administrator Rechte haben bzw. muss Ihnen das root-Passwort zugänglich sein.

e auf dem LINUX Rechner müssen die Programme "pppd" und "chat" installiert sein. Üblicher Weise finden sie diese Programme im Verzeichnis **/usr/sbin**. Mit den Kommandos "**which pppd**" bzw. "**which chat**" können Sie danach suchen. Sind diese Programme noch nicht installiert, so müssen sie nachinstalliert werden. Bei der SuSE Distribution erfolgt dies mit dem Programm "yast" und bei RedHat mit "rpm".

f der LINUX Kernel muss für den PPP-Support eingerichtet und PPP aktiviert sein (PPP heißt: Point to Ponit Protocol). Dies erfolgt üblicher Weise bereits - auf eine entsprechende Anfrage - bei der Installation des LINUX Systems. Erscheint beim ersten Versuch eine Modemverbindung aufzubauen die Meldung *"this system lacks PPP kernel support"* auf dem Bildschirm, so muss das System mit der ppp-Option neu installiert bzw. der Kernel neu übersetzt werden. Details dazu siehe unter: **/usr/src/linux/README**.

So geht's weiter

a zunächst wird von der Kommandozeile aus das Programm "pppd", der *point to point protocol daemon*, gestartet.

b aus dem Programm **pppd** heraus wird daraufhin das Programm "chat" gestartet, welches zur Kommunikation mit dem Modem benötigt wird.

c üblicher Weise kann man sich auf zwei Arten bei seinem Provider authentifizieren:

c1 entweder über ein Skript, welches sich mit dem Server des Providers unterhält (in diesem Skript steht der Reihe nach, welche Antwort unser Rechner auf die Frage des Servers zu geben hat)

c2 oder mittels PAP (Password Authentication Protocol), damit das funktioniert, müssen Sie Ihre User - ID, Ihre Email Adresse und Ihr Passwort in die Datei **/etc/ppp/pap-secrets** eintragen.

d in der Datei **/etc/resolv.conf** muss die IP-Adresse Ihres Nameservers (bzw. Ihrer Nameserver) eingetragen werden. Nameserver wandeln Namen (z.B. **www.tgm.ac.at**) in die entsprechende 12-stellige IP Adresse (hier: **xxx.xxx.xxx.xxx**) um.

e LINUX Puristen loggen sich nun auf zwei Konsolen als "root" ein (Konsolenwechsel mit Alt F1 bis Alt F6). X-Windows Anwender öffnen zwei **xterm**-Fenster und werden dort mit dem Befehl "su" zu "root". Auf der einen Konsole bzw. in einem **xterm**-Fenster starten Sie **tail -f /var/log/messages**. Damit kann permanent beobachtet werden, was auf Ihrem Rechner so vor sich geht.

f in der Datei **/etc/resolv.conf** muss folgende Zeile stehen: "**nameserver xxx.xxx.xxx.xxx**". Die IP Adresse Ihres *name servers* sagt Ihnen Ihr Provider (ggf. müssen hier mehrere Nameserver eingetragen werden).

g in der Datei **/etc/ppp/pap-secrets** muss die Zeile "**username password**" stehen; beides erhalten Sie ebenfalls von Ihrem Provider.

Achtung: Sie sehen, das Passwort steht im Klartext in der Datei "**pap-secrets**". Mit "**ls -l**" stellen Sie fest, ob wirklich nur der user "**root**" Zugriff hat. Mit dem Befehl "**chmod 600 /etc/ppp/pap-secrets**" können ggf. die Zugriffsrechte entsprechend angepasst werden.

Damit wären die Vorarbeiten abgeschlossen, endlich geht's wirklich los

Geben Sie bitte folgende Zeile ein: (die einzelnen Terme werden auf der folgenden Seite erklärt -> an Stelle von "**abc**" setzen Sie bitte Ihren Namen ein, für "**xyz**" ein Kennwort für Ihren Server z.B. "**tgm**") (Die Zeile finden Sie am Fuß dieser Seite.

Was bedeuten die einzelnen Terme im Aufruf?

Nach Eingabe der Befehlszeile und Drücken der Entertaste sollte das Modem zu wählen beginnen. Sämtliche Schritte des Verbindungsaufbaus sollten im Logfenster mitverfolgt werden können.

Wenn alles richtig klappt, ist Ihr PC nun mit dem Internet verbunden. Im chat-Script kann man auch nachsehen, ob die Eingabezeile ggf. noch fehlerhaft ist. Wir können nun auf der ersten Konsole mit Strg Z den **pppd** anhalten und ihn mit "**bg**" in den Hintergrund schicken. Um die Verbindung zu überprüfen, geben wir z.B. "**ping www.tgm.ac.at**" ein. Mit dem Kommando "**killall pppd**" sollte das Modem sofort auflegen.

Für das weitere Vorgehen ist das Funktionieren der o.a. Befehlszeile die Grundvoraussetzung. Im nächsten Schritt soll die Einwahl in das Internet automatisiert werden. Die einfachste Möglichkeit dazu besteht darin, dass man einen Skript zum Aufbau und einen Skript zum Abbau der Verbindung schreibt. Wir erstellen dazu für den User "**root**" ein Unterverzeichnis, z.B. **/usr/sbin**.

Mit einem beliebigen Texteditor erstellen wir nun die Datei "**Internet_on**" die etwa folgenden Inhalt haben sollte:

```
#!/bin/bash
pppd /dev/cua1 connect \
'chat -v "" at\&f1 OK atdtrufnr CONNECT' \
38400 modem crtscts defaultroute \
user abc remotename tgm
```

Die oberste Zeile markiert den Anfang des Skrips, das *detach* wird nun nicht mehr benötigt - **pppd** soll sich nach getaner Arbeit in den Hintergrund begeben. Die Backslashes fungieren nur als Trenner - es könnte alles in eine einzige Zeile geschrieben werden.

Um die Verbindung wieder abzubauen, wird eine zweite Datei, die Datei "**Internet_off**" benötigt. Sie sieht etwa folgendermaßen aus:

```
#!/bin/bash
killall pppd
```

Damit die beiden Dateien auch ausgeführt werden können, ist folgende Eingabe notwendig:

```
chmod u+x Internet_on
chmod u+x Internet_off
```

Dem User "**root**" ist es nun jederzeit möglich mit "**Internet_on**" die Verbindung aufzubauen und mit "**Internet_off**" die Verbindung wieder zu beenden.

Befehlszeile zum Verbindungsaufbau

```
abc:/root #pppd -detach /dev/cua1 connect 'chat -v "" at\&f1 OK atdt rufnummer CONNECT' 38400 modem crts defaultroute user abc remotename xyz
```

Die Programmierung der bash - Shell

Michael Kugler

Unter DOS gibt es die Datei **command.com**, die die Eingaben der Tastatur bzw. von bat-Dateien verarbeitet. Unter Linux gibt es verschiedene Kommandozeilenprogramme, die man *Shell* nennt. Shell-Programme sind einfache Textdateien mit einigen Linux- und/oder **bash**-Kommandos. Nach dem Start eines derartigen Programmes werden die Kommandos der Reihe nach ausgeführt. Genau so wie bei DOS können Parameter übergeben werden und vom Programm ausgewertete werden.

Die bash (born again shell) ist die am meist verbreitete Shell.

Neben der sequentiellen Abarbeitung unterstützt die bash die Shellprogrammierung durch Schleifen und Verzweigungen. Sie ist damit wesentlich flexibler als die Programmierung von Batch-Dateien unter DOS. Ich möchte ein Programm haben, das mir ein geordnetes Directory-listing erstellt (Die zuletzt veränderten Dateien sollen am Ende des Listings erscheinen.)und am Ende eine kleine Zusammenstellung liefert. Aus der man-Page (Aufruf: **man ls**) erfahre ich, dass die notwendigen Optionen **l,r,t** sind. Der einzutippende Befehl lautet: **ls -ltr**. Mit meinem Lieblingseditor (es gibt unter Linux viele derartige !) erstelle ich eine Textdatei mit dem folgenden Inhalt

```
ls -ltr $*
```

und speichere diesen Text unter der Bezeichnung **lt** (so soll mein neues Programm heißen) ab. Die in der **bash** definierte Variable **$***, enthält alle an sie übergebenen Parameter. Es ist also z.B. möglich, sich alle Dateien anzusehen, die die Endung **.txt** haben.

Damit dieser Text zu einem Programm wird, ist es notwendig, den Zugriffmodus auf ausführbar zu schalten.

```
chmod +x lt
```

Um dieses kleine Programm zu testen, gebe ich das folgende Kommando ein.

```
./lt *.txt
```

danach ergibt sich das folgende Listing

```
-rw-r--r--   1 michi    users       70373
Nov 29 14:21
DE-DOS-nach-Linux-HOWTO.txt
-rw-r--r--   1 michi    users       32179
Dez 14 10:37 f2.txt
-rw-r--r--   1 michi    users       31424
Dez 19 11:36 l2d.txt
-rw-r--r--   1 michi    users         369
Dez 19 12:07 bash-programmierung.txt
```

Da das Arbeitsverzeichnis (aus Sicherheitsgründen) kein Bestandteil des Suchpfades ist, muss die Pfadangabe (hier **./**) dem Programmnamen vorangestellt werden. Wenn das Programm dann so funktioniert, wie es soll, kann es in ein Verzeichnis gestellt werden (z.B. **/bin**), welches im Suchpfad enthalten ist.

Das Programm soll auch funktionieren, wenn eine andere Shell dieses Programm aufruft. Zu diesem Zweck wird unserem Programm **lt** in der ersten Zeile mitgeteilt, welche Shell dieses Programm bearbeiten soll. Die erste Zeile in dem Programm lautet:

```
#!/bin/sh
```

Nun zum zweiten Teil der Aufgabe. Das Programm soll uns am Ende des Verzeichnislistings eine Zusammenfassung (die Anzahl der angezeigten Dateien und der gesamte Speicherbedarf)anzeigen.

Dazu wird die Verzeichnisinformation neben der Ausgabe auf dem Bildschirm auch in eine temporäre Datei geschrieben. Den Namen dieser Datei verbinde ich mit der Prozessnummer. (Die Shellvariable **$$** beinhaltet diese Nummer). Da es keinen Sinn macht, diese Datei anzuzeigen, schließe ich sie vom Anzeigen aus. Der neue Befehl für das Anzeigen der Dateien lautet:

```
ls -ltr -I tmp.$$ $* | tee tmp.$$
```

(**-I** schließt die Datei **tmp.$$** von der Anzeige aus, **|tee** bedeutet, dass neben der Anzeige auf dem Bildschirm in die Datei **tmp.$$** geschrieben wird. **$$** wird durch die aktuelle Prozessnummer ersetzt.)

Als Nächstes wird diese temporäre Datei in einer `for`-Schleife bearbeitet. Die beiden lokalen Variablen `summe` und `anzahl` werden vorher auf 0 gesetzt. Das Pro-

pppd	Starten des Point to Point Dämon
-detach	der **pppd** soll zunächst nach dem Aufruf auch weiterhin im Vordergrund arbeiten
/dev/cua1	bezeichnet die serielle Schnittstelle, an der das Modem angeschlossen ist. **/dev/cua0** steht für **COM1** und **/dev/cua1** steht für **COM2**.
connect	steht vor dem Programm, welches mit dem Modem spricht. Hier wollen wir "**chat**" verwenden. Alles was wir **chat** mitgeben wollen, muss unter einfachen Anführungszeichen (') stehen.
'chat -v "" at\&f1 OK atdt rufnummer CONNECT'	**chat** wird gestartet, **-v** bewirkt, dass **chat** alles mitloggt -> mit "**tail -f /var/log/messages**" können wir zusehen, wie die Kommunikation abläuft. Beim Kommunikationsaufbau handelt es sich um ein Frage - Antwortspiel zwischen unserm PC und dem angerufenen Rechner (=Server). Zunächst wartet unser Modem auf nichts (""). Als nächstes wird das Modem mit **at&f** initialisiert, danach waret das Programm auf ein **OK** vom Modem. Danach wird mit Tonwahl die "**rufnummer**" gewählt und auf ein **CONNECT** gewartet.
38400	ist ein Vorschlag für die Kommunikationsgeschwindigkeit zwischen PC und Modem; es sollte hier auch schneller gehen: z.B. 115200. Anmerkung: die Übertragungsrate von Modem zu Modem ist abhängig von der Qualität der Telefonleitung und kann von Anwender nicht beeinflusst werden.
modem	zur Kommunikation verwenden wir ein Modem
crtscts	die Flusssteuerung erfolgt durch die Hardware
defaultroute	die Modemverbindung wird zum Standardweg aller Datenpakete die ins Internet gehen sollen.
user abc	hier muss genau der Username stehen, der in **/etc/ppp/pap-secrets** eingetragen wurde.
remotename xyz	hier muss das zweite Wort aus **/etc/ppp/pap-secrets** (die Kurzbezeichnung für den Server).