Linux Einführung im WS 02/ 03

Jan Feilen

www.feilen.de/linux/

30. September 2002

Inhaltsverzeichnis

1 Vorwort

Ich habe diese Einführung für die Fachschaft Informatik und Mathematik Universität Mannheim geschrieben. Sie soll einen kleinen Überblick über die Grundlegenden Funktionen von Linux geben. Ich erhebe kleinerlei Anspruch auf Vollständigkeit und bestimmt setzen andere Leute auch andere Schwerpunkte. Doch ich habe versucht meine Schwerkunkte so zu setzen, wie ich es aus meiner Erfahung mit Linux und der Uni für richtig halte. Natürlich stehe ich für weitere Fragen gerne zur Verfügung (Email an mich: jan@feilen.de).

2 Was ist Linux?

2.1 Geschichte

Linux ist ein unixartiges Betriebssystem. Es wurde im Jahre 1991 von Linus Torwalds entwickelt und seit dem von einger großen Gruppe von Programmieren weiterentwickelt. Es läuft derzeit auf ca. 12 Millionen Rechner und erfreut sich besonders in letzter Zeit großer Beliebtheit.

2.2 Aufbau von Linux

Linux ist speziell genommen nur der "Kernel". Der "Kernel" ist das "Herz" eines Betriebssystems. Das System wird duch Anwendungen vervollständigt. Der Kernel verwaltet also die Grundlegenden Funktionen des Betriebssystems, so z.B. den Arbeitsspeicher, Festplatten, usw. Auch stellt er eine standartisierte Schnittstelle zwischen den Anwendungen und der Hardware her. Ohne weiter Anwendungen wäre also der Kernel nutzlos. Diese Anwendungen können z.B. Konsolen-Programme sein. Aber auch das "X Window System" baut auf den Kernel auf. Es bietet standartisierte Routinen für grafische Anwendungen. Jedoch ist dies im Gegensatz zu Windows wesentlich "freier". Der sogenannte "X-Server" bestimmt nicht, wie die Grafische-Oberfläche nacher aussieht. Dafür braucht man eine weitere Instanz, einen "Window-Manager". Dieser stellt erst die eigentliche *GUI* (Graphical-User-Interface) her. Verbreitete "Window-Manager" sind z.B. KDE oder Gnome.

2.3 Linux-Distibutionen

Da Linux ein freies Betriebssystem ist, es also keinem Konzern gehört und jeder theoretisch selbst etwas ändern kann, gibt es auch nicht *das eine* Linux.
Das Betriebssystem besteht ja, wie gesagt, aus dem Kernel und weiteren Anwendungen. Deshalb gibt es auch verschiedene Distributionen. Alle haben zwar (in etwa) den gleichen Kernel, aber unterscheiden sich doch teilweise sehr in den Anwendungen.
Manche Distributionen sind vollkommen frei im Netz zu bekommen. So wird z.B. Debian nicht kommerziell vertrieben und man kann das komplette System von einem FTP-Server ziehn. Andere werden kommerziell vermarktet, wie z.B.

SuSe und RedHat. Hier kauft man sich normal die CDs von der jeweiligen Firma. Der Vorteil besteht oft in den mitgelieferten Handbüchern. Es ist allerdings auch erlaubt sich eine solche Distribution einfach von einem Freund zu kopieren - wie gesagt, Linux ist ein freies Betriebssystem. ;-)
Wer sich eine Linux-Distribution auf seinem Rechner installiert, bekommt normal wesentlich mehr also bei einem Windows-System. Da sich die Distributionen ja gerade bei den Anwendungen unterschieden, liegen auch immer sehr viele Anwendungen den Distibutionen bei. So umfaßt z.B. SuSe 8.1 in seiner Professional Ausgabe ganze 7 CDs.

2.4 Wie sind Verzeichnisse unter Linux angeordnet?

Um es vorweg zu sagen: Hier ist wohl alles anders als bei Windows. ;-)
Linux kennt generell keine "Laufwerke". Alles liegt in einer Baum-Struktur vor. So hießt die höchste Ebene, also die Wurzel, nur kurz "/" (gesprochen "root"). Hier werden die einzelnen Festplatten und sonstigen Datenträger hineingelegt (ein Laufwerk einzubinden wird "mounten" genannt. Da es über den Befehl *mount* geschieht). Auch unterscheidet Linux zwischen Groß- und Kleinschreibung. Es kann so z.B. in einem Verzeichnis eine Datei mit dem Namen "fuchur.txt" neben der Datei "Fuchur.txt"existieren.
Um einen kleinen Überblick über die Struktur zu geben, hier kurz, wo sich was befindet:

- /etc/ - hier liegen allgemeine Konfigurationsdateien
- /usr/share/ - Programme
- /usr/doc/ - Dokumentationen
- /var/ - Veränderbare Dateien
- /home/ - unter dem Benutzernamen liegen hier die Heimverzeichnisse der Benutzer

3 Wichtige Befehle

Hier also mal ein paar Befehle die ich für wichtig halte:

- **passwd** - change user password
  Hiermit könnt ihr euer Passwort ändern.
- **exit**
  Hiermit logt ihr euch wieder aus
- **man** - Programm zum Einsehen der Online-Manuale
  man [OPTION]...[befehl]
  Einer der wichtigsten Befehle! Hier bekommt ihr hilfe zu eigentlich allen Fragen.

- **ls** - list directory contents
  ls [OPTION]...[FILE]...
  Hiermit zeigt ihr den Inhalt eines Verzeichnisses an. Optionen sind zum Beispiel:
  - a - listet alle Dateien im Verzeichnis auf, auch versteckte
  - l - listet alles in einem längeren Format auf
  - h - steht für "Human readable", Dateigröße wird in "menschenverständlichen" Größen ausgegeben
  - R - zeigt rekursive alle Verzeichnisse an
- **cd** - change directory
  Wechselt in das angegebene Verzeichnis. Weiter wichtige Optionen sind:
  - ~ - wechselt in des Home-Verzeichnis des Benutzers
  - .. - wechselt in das höherliegende Verzeichnis
- **mkdir** - make directories
  mkdir [OPTION] DIRECTORY...
  Hiermit werden Verzeichnisse erstellt.
- **cp** - copy files and directories
  cp [OPTION]...SOURCE DEST
  Hiermit werden, welch Wunder, Dateien und Verzeichnisse kopiert.
  - R - kopiert rekursiv
  - f - "force", stellt keine lästigen Fragen ;-)
- **mv** - move (rename) files
  mv [OPTION]...SOURCE DEST
  Hiermit werden Dateien und Verzeichnisse verschoben oder auch nur umbenannt.

  Umbenennen einer Datei:
  mv fuchur.txt fuchur.asc
  Hier wird die Datei "fuchur.txt" in "fuchur.asc" umbenannt.
- **rm** - remove files or directories
  rm [OPTION]...FILE...
  - R - löscht rekursiv

  Löschen eines Verzeichnisses
  rm fuchur/ -R
  Hier das Verzeichnis "fuchur" (mit allen Inhalten) gelöscht.

- **w** - Show who is logged on and what they are doing
  Hiermit seht ihr, wer noch alles auf dem System eingeloggt ist und was sie zur Zeit machen. Sehr nützlich, wenn ein System plötzlich langsamer wird ;-)

- **ps** - report process status
  ps [OPTIONS]
  Hiermit seht ihr, welche Prozesse zur zeit auf dem System laufen. Dies ist sehr wichtig, wenn ein Prozess einmal hängengeblieben ist und man in "töten" muß

- **kill** - terminate a process
  kill [-s signal | -p] [-a] [--] pid ...
  Hiermit werden Prozesse beendet.

  Um einen Prozess endgültig abzuschiesen:
  kill -9 pid

- **more** - file perusal filter for crt viewing
  more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] [file ...]
  Hiermit könnt ihr euch den Inhalt einer Datei anzeigen lassen.
  Mit *Space* geht ihr eine Seite weiter und mit *q* beendet ihr *more*.

- **less** - opposite of more
  Ähnlich wie *more*, doch läßt auch das vor und zurück springen in einen Text zu.

- **df** - report filesystem disk space usage
  df [OPTION]...[FILE]...
  Hiermit seht ihr, wieviel Platz noch den Laufwerken ist.
  Nützliche Optionen sind:

  - m - zeigt den Platzenplatz in Megabyte an

- **du** - estimate file space usage
  du [OPTION]...[FILE]...
  Zeigt wieviel Platz z.B. ein Verzeichnis belegt.
  Nützliche Optionen sind:

  - h - "human readable" (siehe oben)
  - s - "summarize", zeigt nur eine Zusammenfassung an
  - a - zeigt Größe für jede Datei an, nicht nur für Verzeichnisse

- **vim** - Vi IMproved, a programmers text editor
  vim [options] [file ...]
  DER Editor unter Linux (und vielen anderen OS'en). *Vim* ist klein und sehr mächtig. Leider würde es diese Einführung sprengen *Vim* zu erklären. Deswegen nur ein paar wenige grundlegende Befehle:

- i - versetzt *Vim* in den *Insert-mode*. Jetzt kann man neue Einträge in die Datei machen. Beendet wird das ganze mit *ESC*.
- :w - schreibt die Datei
- :q - beendet *Vim*. Wenn noch ein "!" angehängt wird, wird Vim ohne nachfragen beendet
- x - löscht das Zeichen, an dem sich der Courser gerade befindet

Der Vollständigkeit muß noch erwähnt werden, daß auch ein großer Teil der Linux-Gemeinde *emacs* als Editor benutzt. Auch *emacs* ist ein sehr mächtiger Editor.

P.S.: Dieses Document wurde mit *Vim* erstellt. ;-)

4 Zugriffsrechte

Einer der vielen Vorteile von Linux gegenüber Windows ist wohl auch die gute Kontrolle von Rechten über Zugriffsrechte. Es kann für jede Datei gesagt werden, wer sie alles öffen, schreiben oder ausführen darf. Jede Datei "gehört" einem Benutzer auf dem System.
Mit *ls -la* kann man sich z.B. die Rechte und den Besitzer einer Datei anzeigen lassen.
Die Ausgabe sieht z.B. so aus:

drwxr-xr-x 10 fuchur users 320 Sep 16 13:03 bin

Die ersten 10 Zeichen, geben die Zugriffsrechte an. Das "d" sagt aus, daß es sich hier im ein Verzeichnis ("Directory") handelt. Die nächsten 3 Zeichen stehen für den Besitzer ("Owner") des Verzeichnisses, die nächsten 3 geben die Rechte für die Gruppe ("Group") an, die letzen 3 für alle Anderen ("All"). Die einzelnen Buchstaben bedeuten:

- r - steht für "read", ob diese Datei überhaupt gelesen werden darf
- w - steht für "write", also schreibzugriff
- x - steht für "execute", ob eine Datein ausgeführt werden darf, bzw. bei Verzeichnissen, ob der Benutzer in dieses Verzeichnis hinein darf.

Danach folgt der Besitzer der Datei und die Gruppe in der er ist (die Datei angelegt hat).
Soweit so gut... und hier jetzt ein paar Befehle, mit denen ihr die Rechte von Dateine und Verzeichnissen ändern könnt:

- **chmod** - change file access permissions
  chmod [OPTION]...MODE
  Mit einem "+" geben wir mehr Rechte und mit einem "-" nehmen wir Rechte.

Ein Beispiel, wir wollen die Datei "fuchur.txt" für "Alle" zugänglich machen:
chmod a+r fuchur.txt
Allerdings nicht für die Gruppe in der wir sind:
chmod g-r fuchur.txt

- **chown** - change file owner and group
  chown [OPTION]... OWNER[:[GROUP]] FILE...
  Mit *chown* ändert man den Besitzer und die Gruppe einer Datei/ Verzeichnis.
  Wir wollen als Beispiel, die Datei "fuchur.txt" an den Benutzer "gerd" aus der Gruppe "kanzler" übergeben:
  chown gerd:kanzer fuchur.txt

5 Die bash

Eine "Shell" ist vergleichbar mit einer "MS-Dos Box", nur wesentlich mächtiger. Die *bash* ist eigentlich die Standart Shell unter Linux, sie ist *sh* kompatibel. Aber genug Tech-Talk, hier ein paar wichtige Besonderheiten:

- & - durch einfaches anfügen eines "&" hinter einem Befehl, wird dieser im Hintergrund gestartet
- *TAB* - autoersetzen von Befehlen und Dateinamen. 2 maliges drücken gibt eine Liste mit Möglichkeiten aus.
- *STRG+r* - durchsucht die bash-History nach demletzt benutzen Befehlen, spart also oft viel tipparbeit.
- ; - trennt Befehle von einander. So kann man Befehle die hintereinander abgearbeitet werden müßen auf einmal eingeben.
- | - *Pipe* genanntes Symbol. Die Ausgabe eines Befehles wird direkt zu dem folgenden "gepiped". So kann man z.B. mit *ps* und *kill* direkt einen Prozess beenden.
- > [FILE] - schreibt die Ausgabe eines Befehls in ein bestimmtes [FILE]

6 ssh - Die Secure Shell

Vielen Leuten dürfte *telnet* ein Begriff sein. *SSH* ist so ähnlich ;-). Der wichtigste Vorteil von *SSH* gegenüber *telnet* besteht darin, daß hier der gesamte Datenverkehr verschlüsselt wird! Es ist also vergleichsweise "sicher" sich von außen einzuloggen. *SSH* bietet allerdings mehr als nur einen remote login. Es ist z.B. möglich mit dem Befehl *scp* Dateien sicher von einem Rechner auf einen anderen zu kopieren. Auch ein sehr großer Vorteil

ist die Möglichkeit das ganze Display auf einen anderen X-Server zu exportieren. Weiterhin bietet *SSH* die Möglichkeit bestimmte Ports weiterzuleiten. Somit kann man z.B. zwischen einem Rechner im Uni-Netzwerk und sich einen verschlüsselten Tunnel aufbauen und Rechner ansprechen, die normalerweise von "außen" nicht erreichbar wären. (Sehr nützlich um z.B. von außen auf die Klausurensammlung der Fachschaft zuzugreifen, die normal nur über das Campusnetzwerk erreichbar ist.)
So, genug der Theorie, hier ein paar Beispiele:

6.1 wichtige Befehle rund um *SSH*

- ssh - SSH client (remote login program)
  ssh [-l login_name] hostname | user@hostname [command]
  Die ist der eigentlich *SSH*-Client. Hiermit werden wir arbeiten.
- scp - secure copy (remote file copy program)
  scp [user@]host1:]file1
  Hiermit werden fast wie bei *cp* Dateien und Verzeichnisse kopiert.

6.2 Einloggen auf einem Server

Erstmal das einloggen auf einem Server. Wir benutzen Hierfür den "rummelplatz". Dort hat jeder Student der Uni Mannheim einen Account. Dieser Rechner ist von außen und von innerhalb der Uni zu erreichen. Er bietet sich damit also an, um ihn als "Einfallstor" zu Rechnern zu benutzen, die man sonst nicht ansprechen könnte. Übrigens könnt ihr auch hier eure EMails checken.
Als Beispiel melden wir den Benutzer "fuchur" auf dem "rummelplatz" an:
ssh fuchur@rummelplatz.uni-mannheim.de

6.3 Portforwarding

Weiterhin soll der Port 11002 des localen Rechners (also vor dem ihr gerade sitzt) durch den "rummelplatz" auf den Port 80 des Fachschaftsservers zeigen:
ssh fuchur@rummelplatz.uni-mannheim.de -L 11002:www.fim.uni-mannheim.de:80
Wenn ihr nun z.B. mit *Mozilla* auf http://localhost:11002 geht, landet ihr auf dem Server der Fachschaft.

6.4 Sicheres kopieren mit *scp*

Wir wollen die Datei "fuchur.txt" in unser Home-Verzeichnis auf den "rummelplatz" kopieren.
scp fuchur.txt fuchur@rummelplatz.uni-mannheim.de:~

6.5 Display umleiten

Um ein Display, also die graphische Ausgabe eines Programms, weiterzuleiten, sollten 3 Sachen gegeben sein:

1. Ihr solltet über einen X-Server verfügen. Diese sind unter Linux Standart, aber auch für andere OS'e erhältlich.
2. Ihr solltet über einen breiten Zugang verfügen. Natürlich richtet sich dies auch nach dem Programm, daß ihr startet, aber Graphik braucht immer relativ "viel" Bandbreite.
3. Auch der Server sollte über einen relativ breiten Anschluß verfügen. ;-)

So, nachdem das alles geklärt ist, kann es losgehn:
Als erstes muß man Rechner auf dem man sich einloggt erlauben, den localen X-Server zu benutzen. Dies geht mit dem Befehl *xhost*. Also z.B.:
xhost +192.168.100.1
Erlaubt dem Rechner mit der IP 192.168.100.1 auf den X-Server zuzugreifen.
Jetzt logt man sich auf diesem Rechner einfach ein, das X-Forwarding ist normalerweise automatisch eingeschaltet:
ssh fuchur@192.168.100.1
Jetzt kommt der eigentliche Trick an der ganzen Sache: Die Variable, die für die Auswahl des Displays zuständig ist, muß man auf unsern Rechner zeigen. Dies geht so:
export DISPLAY=192.168.100.12:0
Hiermit haben wir das Display auf das 0. Display (also die 1. Grafikkarte) unseres Rechners (mit der IP 192.168.100.12) gesetzt.
FERTIG! Jetzt einfache ein Programm starten, je nach Geschwindigkeit erscheint das Fenster mehr oder weniger schnell. ;-)

7 Schlußwort

So, das war es auch schon. Ich hoffe auf der einen Seite, daß es verständlich war und auch auf der anderen Seite, daß es nicht zu viel war. Wenn man viel mit Linux arbeitet verliert man oft aus dem Blick, wo ein Anfänger Probleme haben könnte.
Auch hoffe ich, daß es Spaß gemacht hat. Ich liebe dieses System und möchte es nichtmehr missen ;-). Es kostet jeden am Anfang sehr viel Zeit sich in die Strucktur von Linux zu gewöhnen, aber ich kann nur sagen, daß es sich lohnt diese Zeit zu investieren.
Also: happy hacking!