# IF TIF 04: Linux ist weiblich

**Ein Kurs im Sommerstudium der Informatica Feminale 2005 in Bremen**

02.–04. September 2005

Dozentinnen: Patricia Jung, Gabriele Pohl

Informatica-Kursskript „Linux ist weiblich“
5. Auflage 2005
Diese Skript stellt eine verbesserte, erweiterte und überarbeitete Fassung der ursprünglich für die Informatica Feminale 2001 erarbeiteten gleichnamigen Kursunterlage von Sibylle Nägle und Patricia Jung dar.

Satz: TEX/LATEX (KOMA-Skript) unter Linux

# Inhaltsverzeichnis

# Intro

Herzlich willkommen zum fünften „Linux ist weiblich“-Kurs in Folge auf der Informatica Feminale! Wie die Ankündigung sagt, richtet er sich an Anfängerinnen und setzt keine tieferen Betriebssystemkenntnisse von anderen Rechnerplattformen voraus, wohl aber Erfahrung im allgemeinen Umgang mit Computern. Ziel ist es, prinzipielle Konzepte von Linux/Unix zu vermitteln und den ersten Einstieg zu erleichtern. Es soll uns dabei jedoch nicht darum gehen, einen Geschwindigkeitsrekord im Erlernen von Unixbefehlen aufzustellen.
Das vorliegende Skript ist nicht als Lehrbuch gedacht[1], sondern eher als Gedankenstütze für die Kursnachbereitung. Die behandelten Aspekte werden oft nicht erschöpfend dargelegt, sodass sich zusätzliche Notizen empfehlen. Ohne die Ausführungen im Kurs und den entsprechenden praktischen Übungen am Rechner mögen Teile des Skripts unverständlich erscheinen oder Vorkenntnisse voraussetzen, die die Nur-Leserin u. U. nicht hat.
Allerdings hoffen wir, dass wir während der drei Tage genügend Grundlagen legen, sodass sich Abschnitte, für die keine Zeit mehr war, im Eigenstudium nacharbeiten lassen. Dass das Skript mehr Stoff ausgearbeitet enthält, als wir vermutlich schaffen werden, war eine bewusste Entscheidung: So haben wir hoffentlich die Möglichkeit, individuellen Präferenzen Rechnung zu tragen. Das heißt aber auch, dass die im Skript angelegte Gliederung nach Tagen nur als Vorschlag und keineswegs als strikter Plan zu sehen ist!
Da es keine Scheine gibt, die die Leistungen der einzelnen Teilnehmerin benoten, können wir ganz entspannt im Team arbeiten. Bitte nutzt diese Möglichkeit bei der Lösung der Aufgaben! Die Mailingliste `tif05@lists.answergirl.de` bietet eine Plattform, auf der Aufgaben diskutiert und gemeinsam Lösungen gefunden werden dürfen.
Unserer Erfahrung nach merkt sich frau Dinge besser, die sie selbst ausprobiert hat. Wer die Antwort auf eine Übungsfrage nicht im Alleingang findet, sollte eine gemeinsam gefundene Lösung daher auf jeden Fall selbst praktisch ausprobieren.
Bei Fragen stehen wir, soweit es die Zeit erlaubt, persönlich, aber auch auf der Mailingliste zur Verfügung. Bitte denkt beim Fragen daran, dass auch andere Kurs-Teilnehmerinnen am selben Problem interessiert sein können, und gebt Eure Erkenntnisse (sofern Ihr sie nicht ohnehin aus der Mailingliste bezogen habt) weiter!
Die Vorkenntnisse der Kursteilnehmerinnen sind sehr unterschiedlich und verlangen daher von den Fortgeschritteneren eine gewisse Rücksicht. Sollte sich die eine oder andere jedoch über mehrere Abschnitte unterfordert fühlen, lasse sie uns das bitte wissen; sicher finden wir eine Lösung. Wenn Ihr jedoch alles schon könnt, was dieses Skript zu vermitteln sucht, seid Ihr leider im falschen Kurs...
Wir freuen uns auf gute Zusammenarbeit, viel Spaß und den einen oder anderen Geistesblitz

Gabriele und Patricia

[1] In einigen Abschnitten enthält es – mittlerweile stark überarbeitetes – Material aus einem von Patricia Jung und Detlev Degenhardt 1997 verfassten Skript für Unix-Kompaktkurse am Rechenzentrum der Uni Freiburg.

# Freitag, 02. September 2005

## 1 Begriffsverwirrung

Das vermutlich größte „Problem“ beim Erstkontakt mit Linux besteht darin, dass es sich nicht unbedingt mit dem Vokabular beschreiben und in die Kategorien einordnen lässt, die frau vielleicht von anderen Betriebssystemen her kennt. So dreht sich eines der am weitesten verbreiteten Missverständnisse um die Frage, was Linux eigentlich ist: Wenn wir heute von Linux, Windows etc. sprechen, meinen wir meist weitaus mehr als das pure Betriebssystem selbst. Wir beziehen oft „alles, was an Software dabei war“ mit in die Definition ein. Das ist verständlich, denn mit dem Betriebssystem selbst, dem *Kernel*, kann die normale Anwenderin überhaupt nichts anfangen ist, sie kann noch nicht einmal auflisten, welche Dateien es auf dem Rechner gibt.
Damit daraus überhaupt eine für Anwendungszwecke brauchbare Software-Umgebung wird, braucht es mehr, und um dieses Mehr kümmern sich bei Linux als Gemeinschaftswerk einer übers Netz verteilten Entwickler(innen)schar und Open-Source-Projekt eben nicht die Kernel-Maintainer.[2]
Stattdessen herrscht Arbeitsteilung: Das Zusammenstellen und Vorkonfigurieren von Kernel und „übriger“, zum Großteil ebenfalls freier Software nennt frau eine *Distribution*. Davon gibt es diverse: Firmen, die als Linux-Distributor auftreten, lassen sich ihre Mühe meist in Form eines Kaufpreises vergüten. Viele davon fühlen sich der Open-Source-Community verpflichtet. Das äußert sich zum einen darin, dass frau die meisten dieser Distributionen (in der Regel abzüglich kommerzieller Addons) aus dem Netz herunterladen kann. Zum anderen stehen tragende Entwickler[3] verschiedener Open-Source-Projekte bei Distributoren auf der Gehaltsliste, und treiben die entsprechende Entwicklung voran.
Diverse freie Projekte stellen ebenfalls Distributionen zusammen. Einige davon sind ihren kommerziellen Schwestern in technischen Detailfragen sogar überlegen, allerdings lässt sich pauschal (fast immer noch) sagen, dass kommerzielle Distributionen die ausgefeilteren und nutzerfreundlicheren Installationsroutinen mitbringen. Dafür bieten die freien Distributionen oft bessere Mechanismen, um das System auf dem aktuellen Stand zu halten, wobei die kommerziellen Distributoren hier in den letzten Jahren teils durchaus aufgeholt haben. Eine Domäne der freien Projekte sind hingegen Spezialdistributionen für bestimmte Einsatzzwecke, seien es Rettungssysteme, Distributionen für Router, Multimedia-Distributionen oder solche für den Einsatz in bestimmten Sprachregionen.
Zwei gegenläufige Entwicklungen ließen sich in den letzten Jahren beobachten: Zum einen ziehen sich die großen kommerziellen Distributoren sowohl im Server- als auch im Desktop-Bereich mehr und mehr in den „Business“-Bereich zurück. Teils entwickeln sie zwar noch eine Distribution für Endverbraucherinnen, doch dient diese de-facto als Testwiese für Businessprodukte. Letztere basieren dann auf der vorangegangenen, fehlerbereinigten Enduser-Distribution. Zum anderen rufen Firmen eigene freie Distributionsprojekte ins Leben, denen sie Unterstützung zukommen lassen und deren Arbeit dann als Grundlage der Businessprodukte dient, teils setzen

[2] Frauen sind hier leider mehr als nur unterrepräsentiert. Val Henson (http://infohost.nmt.edu/~val/), die als Kernel-Entwicklerin für IBM tätig ist, gehört zu den wenigen Ausnahmen.
[3] Frauen, die in diese Kategorie (Betonung auf „tragend“) fallen, sind uns hier leider nicht bekannt.

sie auf existierenden freien Distributionen auf und „beschränken“ ihre Arbeit auf Eigen- und Weiterentwicklung in Bereichen, in denen die freie Distribution Defizite hat.

Eine einheitliche Installationsroutine für alle Distributionen gibt es nicht – gerade sie stellt ein wesentliches Alleinstellungsmerkmal der einzelnen Distributionen dar. Auch bei der Wartungssoftware stößt frau oft auf distributionsspezifische Eigenentwicklungen.

Diese „Zersplitterung“ in Distributionen hat Vor- und Nachteile: Der Vorteil besteht darin, dass frau sich genau die heraussuchen kann, die ihren Bedürfnissen am ehesten entspricht. Ein Nachteil: Frau muss u. U. eine Menge Frösche küssen, ehe sie die Prinzessin findet. Ein wahrscheinlich noch schwerwiegenderer: Was bei der einen Distribution so funktioniert, kann bei einer anderen komplett anders aussehen. Daher ist es gut, darüber Bescheid zu wissen, wie ein Linuxsystem an sich aufgebaut ist, statt das eigene Wissen an Tools einer bestimmten Distribution festzumachen. Allerdings gibt es auch hier in Konfigurationsdetails Unterschiede, und unter den kommerziellen Distributionen findet sich mittlerweile kaum noch eine, die den Original-Kernel (auch „Vanilla-Kernel“ genannt) verwendet ...

Wenn Linuxerinnen Hilfe suchen, ist es deswegen sehr wichtig, dass sie dazusagen, welche Distribution in welcher Version sie verwenden. Außerdem sollten sie niemals den Fehler begehen, die Distributionsversion mit der Kernelversion zu verwechseln: Im besten Fall schüttelt die Helfende nur den Kopf über die arme Newbie, kann sich aber denken, was diese gemeint hat. Im schlechteren Fall gibt es mehrere Distributionen, auf die die genannte Version passen könnte. Im schlimmsten Fall bekommt die Fragende einfach nur die vielleicht patzige, aber durchaus korrekte Antwort: „Linux x.y.z gibt es nicht.“

✎ Die in Mittel- und Westeuropa zur Zeit am weitesten verbreiteten Linux-Distributionen sind *SuSE Linux* (aus dem Portfolio der Firma Novell), *Fedora Core* (ein freies Projekt, das die Firma Red Hat ins Leben rief, unterstützt und als Basis der eigenen Businessprodukte nutzt[4]), *Mandriva Linux* (die aus der Übernahme des brasilianischen Distributors Conectiva durch die französische Firma Mandrakesoft entstandene kommerzielle Distribution) sowie *Debian GNU/Linux*. Finde die aktuellen Versionen dieser vier Distributionen heraus!

✎ Suche drei Distributionen heraus, die auf Debian aufsetzen!

✎ Du bekommst eine Support-Mail mit dem Hinweis „Ich benutze Linux 9.3.“ Was könnte die Fragende gemeint haben?

Außer den bekannten Distributionen für den Desktop- und Servereinsatz gibt es Unmengen Zusammenstellungen für spezielle Einsatzzwecke, angefangen von der Videorekorder-Distribution *LinVDR* bis hin zu *Skolelinux*, einer Debian-basierten Lösung für Schulnetze. Einen Boom erlebten in letzter Zeit die Live-Distributionen, mit denen sich ein Linux-System von CD oder DVD betreiben lässt, allen voran das ebenfalls auf Debian aufbauende *Knoppix*. Knoppix eignet sich dank seiner guten Hardware-Erkennung gut zum Testen, ob ein potentiell zu erwerbender Rechner Linux-tauglich ist, und dient selbst wiederum als Basis für viele andere abgeleitete Distributionen.

## 2 Komponenten

Eine der wichtigsten Eigenschaften von Unix-Systemen ist ihre Flexibilität. Statt großer Anwendungspakete und komplizierter Dienstprogramme spielen in der Unix-Philosophie kleine Programme eine wichtige Rolle, die jeweils lediglich eine Aufgabe (perfekt) erfüllen, sich aber zu mächtigen Werkzeugen zusammensetzen lassen. Am deutlichsten sichtbar wird dieses „Baukastenprinzip“ auf der Kommandozeile.

Es gilt auch im Großen: Frau kann sich im Prinzip ihr eigenes Linuxsystem von Grund auf selbst zusammensetzen. Das kostet lediglich Online-Gebühren und eine Menge Arbeitsstunden – und geht dank „Linux from Scratch“ sogar mit Baukasten und Anleitung.

## 2.1 Kernel

Die wichtigste – daher auch namensgebende[5] – Komponente eines Linux-Systems ist der Betriebssystemkern oder „Kernel“. Er sorgt nicht nur für das Ausführen und Verwalten von Prozessen, sondern stellt auch die Treiber für Dateisysteme, Hardware etc. bereit. Er ist dafür verantwortlich, dass wir es bei Linux mit einem *Multiuser/Multitasking*-Betriebssystem zu tun haben.

**Was bedeutet Multiuser/Multitasking? Ein Ausflug in die Geschichte:** Das altgediente (und immer noch nicht völlig ausgestorbene) DOS ist ein *Singleuser/Singletasking*-System. Einen DOS-Rechner kann immer nur eine Benutzerin (*user*) gleichzeitig bedienen, diese wiederum hat keine Möglichkeit, mehr als eine Aufgabe (*task*) bzw. ein Programm zur gleichen Zeit auszuführen. (Eine Ausnahme macht `print`, das auch im Hintergrund laufen kann.)
Windows 3.x selig war ebenfalls ein Singleuser-System. Hier konnten zwar schon mehrere Programme gleichzeitig arbeiten, aber nur in Form des nichtpräemptiven Multitaskings (to pre-empt – engl.: vorwegnehmen): Auf keinem Rechner mit nur einem Prozessor (egal, welches Betriebssystem) laufen mehrere Programme wirklich gleichzeitig. Der Trick besteht darin, jedem davon unmerklich für die Benutzerin kurzfristig Rechenzeit zur Verfügung zu stellen und zwischen diesen Prozessen sehr schnell hin- und herzuschalten.
Bei Windows 3.x war nun nicht das Betriebssystem für dieses Umschalten (*scheduling*) verantwortlich. Stattdessen mussten die einzelnen Applikationen ihre Rechenzeit von sich aus wieder freigeben („nicht präemptiv“). Dies taten sie häufig nicht, was die legendären „Hänger“ verursachte.
Auf Microsoft-Seite ging Windows 9x hier endlich einen Schritt weiter: Das Betriebssystem selbst sorgt für das Scheduling (also: präemptives Multitasking). Aber auch Win 9x sind immer noch Singleuser-Betriebssysteme. Zudem nehmen sie es mit dem *Speicherschutz* nicht so eng: Wo es nur eine Benutzerin gibt, die alles darf (sogar am Betriebssystem vorbei direkt auf die Hardware zugreifen), ist es dem Betriebssystem selbst unmöglich, User-Prozessen weniger Rechte zu geben als Systemprozessen. Wenn Anwendungsprogramme nicht ganz sauber programmiert sind, können sie sich so beim Belegen des Hauptspeichers in die Quere kommen.
Hier zeigt sich die historische Stärke von Unixsystemen: Auf ihnen können bereits seit vielen Jahrzehnten mehrere Benutzerinnen mehrere Programme gleichzeitig ausführen. Dass Prozesse nicht einfach Speicherseiten anderer Prozesse überschreiben dürfen, ist ebenfalls seit langem selbstverständlich.

[5] Tatsächlich geht es bei einem beliebten Streitthema in der „Linux-Gemeinde“ darum, wie Linux-Systeme benannt werden sollen. Hier argumentieren vor allem die Vertreter der „Free Software Foundation“, dass es „GNU/Linux“ heißen sollte, weil der Linux-Kernel ohne die GNU-Tools unbenutzbar wäre (siehe 16). Das wiederum ruft die Vertreter/innen anderer wichtiger Open-Source-Projekte auf den Plan, die zu Recht sagen, dass man ihren Beitrag dann konsequenterweise ebenfalls durch Namensnennung der Art „Apache/GNU/KDE/MySQL/.../Linux“ würdigen müsste, was an Lächerlichkeit kaum zu überbieten wäre. Dementsprechend gibt es nur einige Projekte wie Debian, die von „Debian GNU/Linux“ etc. sprechen; im allgemeinen Sprachgebrauch heißt auch das Gesamtkunstwerk weiterhin „Linux“.

**Multiuser/Multitasking (Fortsetzung)** Gleichzeitig nimmt sich das Benutzerinnenkonzept (verglichen z. B. mit Windows NT ff.) recht einfach aus (wenngleich es – ohne Erweiterungen – weniger feingliedrig ist), sodass sich die administrative Nutzung des Systems und das Arbeiten als Anwenderin in der Praxis leichter trennen lässt – mit positiven Folgen für die Systemsicherheit.

Allerdings umfasst Linux selbst inzwischen unheimlich viel Funktionalität (kein Wunder bei einem Betriebssystem, das vom Großrechner bis hin zur Armbanduhr auf aller möglichen Hardware läuft). Wer keine SCSI-Geräte hat, braucht z. B. auch keinen Treiber dafür im System. Da passt es gut, dass der Kernel im Quelltext erhältlich ist und frau ihr ganz persönliches Betriebssystem konfigurieren und kompilieren kann.
Dazu benötigt sie das Quelltextpaket, das normalerweise nach `/usr/src/linux` installiert wird. In diesem Verzeichnis findet sich eine Datei namens `README`, die (auf Englisch) erklärt, wie sich frau ihren eigenen Kernel bäckt.
Grundsätzlich ist es für einen Kernel heutigen Ausmaßes keine gute Idee, ständig sämtliche Funktionalität im Speicher zu halten, auch wenn sie gar nicht gebraucht wird (z. B. weil das CD-ROM-Laufwerk gerade nicht in Gebrauch ist). Daher kann frau Treiber, die beim Booten nicht benötigt werden, gut und gern in „ladbare Module“ (*loadable modules*) auslagern, anstatt sie alle fest in einen *monolithischen Kernel* einzukompilieren. Diese werden erst bei Bedarf zum Kernel in den Speicher geladen. Alle bekannteren aktuellen Distributionen spielen einen solchen modularen Kernel ein, sodass das Selbstkompilieren eines ans eigene System angepassten Kernels heute in der Regel nur noch aus zwei Gründen nötig ist:

1. Frau will eine aktuellere Kernel-Version einspielen, als sie der Distributor zur Verfügung stellt.
2. Frau braucht Funktionalität, die der offizielle Kernel nicht enthält und die der Distributor nicht eingepflegt hat. Dann ist Kernelflicken mit Hilfe sogenannter *Patch-Dateien* angesagt.

**Kernelbäckerei** Ein `make clean` säubert die Kernel-Sourcen von Objektdateien etc. voriger Kernel-Kompilieraktionen. Mit `make menuconfig` kann frau den Kernel jetzt nach Gutdünken konfigurieren. Wenn sie auf einer grafischen Oberfläche arbeitet, gibt es alternativ die Variante `make xconfig`, ab Kernel 2.6 zudem `make gconfig`.[6]
Sollte es bei beiden Fehlermeldungen aufgrund fehlender Bibliotheken für die grafische oder pseudografische Ausgabe geben, geht auch `make config`. Die hier zum Einsatz kommende Routine fragt der Reihe nach alle Einstellungen ab, erlaubt aber kein Zurückgehen und ist daher nur im Notfall zu empfehlen – oder dann, wenn frau schon über eine funktionierende Kernelkonfiguration verfügt und nur noch die Punkte abgefragt werden will, die in einer neuen Kernelversion hinzugekommen sind. Dann nutzt sie den Befehl `make oldconfig` und landet in derselben „Abfrageschleife“.

[6] Erhält frau bei einer der drei (pseudo-)grafischen Varianten statt einer Benutzungsoberfläche nur Fehlermeldungen, lässt sich `make menuconfig` erfahrungsgemäß am schnellsten wieder flottmachen: Hier müssen in der Regel nur die Pakete `ncurses` und `ncurses-dev(el)` o. ä. nachinstalliert werden.



# Index

**Kernelbäckerei (Fortsetzung)** Zu den meisten Punkten bekommt frau (mit (?)) Hilfe und eine Empfehlung, ansonsten gilt es, (Y), (N) oder (M) (für Module) zu sagen.
Die Konfiguration wird (so von der Konfigurierin nicht anders festgelegt) in der Datei `/usr/src/linux/.config` abgelegt (statt `linux` kann das Verzeichnis auch `linux-`*Versionsnummer* o.ä. heißen). Ein `make dep` schaut bei älteren Kerneln nach, ob alle Abhängigkeiten erfüllt sind; ab Version 2.6 ist dieser Schritt nicht mehr nötig. `make` (bei Kernelversionen kleiner als 2.6 `make bzImage`) kompiliert den neuen Linux-Kernel, der dann unter `/usr/src/linux/arch/i386/boot/bzImage` liegt und installiert werden kann.
Wird der Bootloader LILO benutzt, so sorgt `make install` (alt: `make bzlilo` statt `make bzImage`) für die Neukonfiguration des Bootloaders; es reicht aber auch, die Datei `bzImage` aus dem `arch/i386/boot`-Verzeichnis unter Erfindung eines sprechenden Dateinamens nach `/boot` zu kopieren, die LILO-Konfigurationsdatei `/etc/lilo.conf` anzupassen und den Bootsektor mit dem Kommando `/sbin/lilo` neu zu schreiben. Wichtig dabei: Niemals den bislang benutzten, funktionierenden Kernel überschreiben oder aus der Bootloader-Konfiguration entfernen, bevor frau nicht sicher ist, dass der neue Kernel auch tut. Dieser Hinweis gilt natürlich auch, wenn ein anderer Bootmanager, zum Beispiel Grub, zum Einsatz kommt.
Dann gilt es noch, die Kernel-Module mit `make modules` zu kompilieren und mit `make modules_install` zu installieren. Bei 2.6er Kerneln entfällt der „`make modules`"-Schritt.

✎ Welche Linux-Versionen sind momentan aktuell? Beachte, dass zum einen (im Gegensatz zu anderen Betriebssystemen) auch ältere Kernel-Serien weiter gepflegt werden. Zum anderen unterscheidet frau zwischen *Entwickler-Kerneln* mit ungerader (Minor-)Versionsnummer und für den stabilen Einsatz gedachten „geraden" Versionen.

✎ Finde einen Kernel-Patch! Für welchen Kernel ist er gedacht?

✎ Sind auf dem Arbeitsrechner die Kernel-Sourcen installiert? Wenn ja: welche Version? Finde einen Punkt heraus, der als Modul konfiguriert ist! Woran hast Du das erkannt?

✎ Finde heraus, welche Linux-Version auf dem Rechner läuft! Benutze dazu die Befehle `uname -a` und `cat /proc/version` auf der Kommandozeile!

## 2.2 Die Shell

Die meisten Betriebssystembefehle sind nichts anderes als C-Programme, die über die sogenannten *System Calls* auf den Kernel zugreifen. Da frau also nicht direkt mit dem Kernel „sprechen" kann, stellt sich die Frage, wie sie mit ihm kommuniziert.
Hier gibt es auf Unix-Systemen zwei Interaktionsmöglichkeiten: über Programme auf einer grafischen Oberfläche und über die klassische Kommandozeile. Auf letzterer läuft die Kommunikation zwischen Benutzerin und dem Betriebssystem immer über einen Kommandointerpreter, eine *Shell*:

Benutzerin ⟺ Shell ⟺ Betriebssystemkern ⟺ Hardware

Das ist übrigens bei DOS genauso (hier heißt die Shell `command.com`), und auch bei Windows XP/2000 gibt es sie.[7] Eine Shell hat die Aufgabe, Eingaben anzunehmen, eventuell ein wenig zu verarbeiten und an die entsprechenden Stellen weiterzuleiten.
Der Name („Muschel", „Schale") kommt daher, dass dieses Programm den Kern des Betriebssystems umschließt und die Kommunikation zwischen Kernel und Benutzerin ermöglicht. Letztere „sieht" also nur die Shell und nicht das Betriebssystem. Das ist wichtig zu wissen, weil es auf ein und demselben System verschiedene Shells gibt, die unterschiedliche Funktionen bereitstellen. Frau kann sie sich im allgemeinen aussuchen, je nachdem, welche Shell dem eigenen Arbeitsstil am meisten entgegenkommt. Allerdings sind nicht immer alle auf jedem Unix-Rechner vorhanden.
Die Shell meldet mit dem *Prompt*, dass sie für Eingaben bereit ist. Typischerweise ist das Prompt-Zeichen ein `$`, ein `>` oder ein `%`. Auch Rechnernamen und Verzeichnisnamen können angezeigt werden (Frau kann das einstellen.):

```
[trish@lillagroenn skript]$
```

Üblicherweise tippt frau einen Befehl auf der Tastatur. Über den Keyboard-I/O-Port (I/O steht für „Input/Output", Ein-/Ausgabe) bekommt der Kernel das mit und schreibt die entsprechenden Zeichen „auf den Bildschirm". Gleichzeitig übergibt er sie an die Shell. Diese bereitet die Eingabe auf, sucht nach einem entsprechenden Programm, bittet den Kernel via *System Call*, es auszuführen, und wartet auf seine Beendigung. Während dieser Zeit übernimmt das aufgerufene Programm die Kontrolle und kommuniziert gegebenenfalls mit der Benutzerin (z. B. durch eine Datenausgabe). Endet das Programm, erhält die Shell die Nachricht, dass sie nun wieder Eingaben annehmen kann.
Unter Linux kommt meist die `bash` („Bourne Again Shell"), eine Neu- und Weiterentwicklung der „Bourne Shell" `sh`, zum Einsatz, die wir im Kurs benutzen. Viele Konzepte lassen sich auch auf andere Shells übertragen. Bei aller Verschiedenartigkeit sind sich Shells auf der Grundlage der Linux-Dateisysteme in einem einig: Auf Unix-Betriebssystemen muss frau auf Groß- und Kleinschreibung achten!

✎ In der Datei `/etc/passwd` steht auf jeder Zeile hinter dem letzten Doppelpunkt, welche Shell beim Login für die jeweilige Benutzerin gestartet wird (*Login-Shell*). Was außer `/bin/bash` ist da zu finden? Zum Anzeigen der Datei kannst Du das Programm `more` verwenden.

✎ Finde zwei Beispiele für andere Shells!

## 2.3 Die Benutzerinnen

Bevor eine Benutzerin überhaupt mit einer Shell arbeiten kann, muss sie einen Zugang dazu bekommen. Zu diesem Zweck verpasst ihr die Systemadministratorin einen Usernamen, der auf dem System – genau genommen in der Datei `/etc/passwd` – eingetragen wird. Diesem Namen ist eindeutig eine UserID (*UID*) zugeordnet. Da Usernamen nur bedingt ein Geheimnis sind, wird der Zugang zudem mit einem Passwort gesichert.
Beide Angaben macht frau am sogenannten *Login-Prompt* des Rechners:

```
Linux Mandrake release 7.0 (Air)
Kernel 2.2.18 on an i686
```

[7] Sie ist hier über die Registry auf User-Basis konfigurierbar.





### Dateisystem und Dateibaum

- *Ext3fs*: http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html
- *Linux Standard Base*: http://www.linuxbase.org/
- *Filesystem Hierarchy Standard*: http://www.pathname.com/fhs/

### Benutzerinnen und ihre Rechte

- *RSBAC*: http://www.rsbac.org/

### Drucken

- http://www.linuxprinting.org/

### Editoren

- `vim`: http://www.vim.org/
- `elvis`: http://elvis.the-little-red-haired-girl.org/
- `pico`: http://www.washington.edu/pine/
- `joe`: http://sourceforge.net/projects/joe-editor/
- `nedit`: http://www.nedit.org/
- `kwrite` und `kate`: http://kate.kde.org/
- `gedit`: http://gedit.sourceforge.net/
- `jove`: ftp://ftp.cs.toronto.edu/cs/ftp/pub/hugh/jove-dev/

### Benutzerinnenverwaltung

- Crack:
  http://www.crypticide.com/users/alecm/security/c50-faq.html

### Installieren von Software

- Checkinstall: http://asic-linux.com.mx/~izto/checkinstall/

### Geschichte

- *Die GNU/Linux-Chronik*:
  http://www.kefk.net/Linux/Chronik/
- *Geschichte freier Software*:
  http://www.kritische-informatik.de/fshistl.htm

### 15.6 Zu den Kursteilen

#### Distributionen

- *SuSE*: http://www.novell.com/linux/suse/
- *OpenSuSE*: http://www.opensuse.org/
- *Debian*: http://www.debian.de/
- *Red Hat*: http://www.redhat.de/
- *Fedora*: http://fedora.redhat.com/
- *Mandriva* http://www.mandriva.com/
- *LinVDR*: http://www.linvdr.org/
- *Skolelinux*: http://www.skolelinux.org/
- *Knoppix*: http://www.knopper.net/knoppix/
- Distrowatch (Übersicht über Distributionen): http://distrowatch.com/
- *Linux From Scratch*: http://www.linuxfromscratch.org/

#### Kernel

- Kernel selbst kompilieren: http://ahrlug.dyndns.org/tip2_kernel26.html
- Val Hensons Tutorial zum Kernel-Hacking: http://www.linuxchix.org/content/courses/kernel_hacking/

#### X-Server

- X.org: *http://www.x.org/*

#### Windowmanager

- EvilWM: *http://evilwm.sourceforge.net/*
- FVWM2: *http://www.fvwm.org/*
- Enlightenment: *http://enlightenment.org/*
- (Nicht vollständige) Übersicht: *http://www.plig.org/xwinman/*

#### Desktopumgebungen

- *KDE*: http://www.kde.org/
- *GNOME*: http://www.gnome.org/





```
lillegroenn login:
```

Nach dem Einloggen (Usernamen und Passwort eingeben; dabei erscheint das Passwort nicht auf dem Bildschirm) öffnet das System die *Login-Shell*.

✎ Finde Deine UserID mit dem Kommando `id` heraus!

✎ Wo steht die UserID in der `/etc/passwd`?

Jede neue Benutzerin wird beim Anlegen zugleich einer oder mehreren *Gruppen* angegliedert. Neben ihrem Gruppennamen hat eine solche Gruppe eine *GroupID*.

✎ Finde mit dem Kommando `groups` heraus, welchen Gruppen Du angehörst!

Damit in einem Multiuser/Multitasking-Betriebssystem nicht das große Chaos ausbricht, muss genau geregelt sein, welche Benutzerin was darf. Daher bekommt jede Datei und jedes Verzeichnis einen Satz von Zugriffsrechten (*permissions*) zugeteilt. Diese regeln, welche Benutzerin welche Aktionen mit der jeweiligen Datei ausführen darf. Mögliche Aktionen sind Lesen (read), Schreiben (write) und Ausführen (execute).
Diese Rechte werden getrennt vergeben für

- die Besitzerin einer Datei/eines Verzeichnisses,
- die Mitglieder der Gruppe, der die Datei zugeteilt wird, und
- alle anderen Benutzerinnen.

Eine Ausnahme bilden lediglich User mit der ID 0. In der Regel ist das nur eine Benutzerin namens `root`, hinter der sich die Systemadministratorin versteckt, die alles darf.

✎ Welchen Gruppen gehört `root` an?

Um Unberechtigten nicht leichtfertig den Zugriff auf den Rechner zu ermöglichen, darf frau nicht vergessen, sich beim System abzumelden, wenn sie mit der Arbeit fertig ist. Dazu dient z. B. das Kommando

```
[trish@lillegroenn skript]$ exit
```

oder die Tastenkombination Strg+d, wenn frau von einer *virtuellen Konsole* aus arbeitet oder von einer Kommandozeile aus auf einem entfernten Rechner eingeloggt ist. Grafische Benutzungsoberflächen bieten in der Regel einen entsprechenden Menüeintrag.
Achtung: Wenn frau die grafische Oberfläche *nach* dem Einloggen selbst gestartet hat, muss sie sich nach dem Beenden derselben extra ausloggen. Frau ist erst dann abgemeldet, wenn sich das System wieder mit dem Login-Prompt meldet!

### 2.4 X-Server

Im Gegensatz zu anderen Betriebssystemen ist die grafische Oberfläche bei Linux u. a. Unixbetriebssystemen nicht ins Betriebssystem integriert. Auf diese Art und Weise lassen sich Linuxrechner komplett ohne den Overhead eines GUIs installieren und benutzen – besonders für

Maschinen sinnvoll, die als Server, Router oder Firewall ihren Dienst tun. Auch ältere i386er können so noch als Text-Terminal betrieben werden.
Um grafisch zu arbeiten, gibt es zwei Alternativen: die (speziell auf Nicht-Intel-Architekturen – Macs, PDAs, ...– verbreitete, aber auch auf PCs laufende) *Framebuffer-Konsole* oder das *X-Window-System*, ein Client-Server-System, das älter als Linux ist und auf den meisten Unices eingesetzt wird. Unter Linux hatte bislang die Open-Source-Implementation des *XFree86*-Teams die größte Verbreitung, das jedoch vor nicht allzu langer Zeit mit lizenztechnischen Querelen auf sich aufmerksam machte. Auf der letzten, der alten „XFree86 Project License 1.0“ unterliegenden XFree-Version 4.4 RC2 basiert die Weiterentwicklung des X.org-Projekts, die sich inzwischen bei den meisten Distributionen durchsetzt.
Hinter dem X-Window-System steckt folgendes Modell: Ein *X-Server* läuft auf einem Rechner und verwaltet die Grafikhardware, damit *X-Clients*, also grafische Anwendungen, darauf zugreifen können. Der X-Server ist quasi der „Treiber“ und muss in der Lage sein, die Grafikkartenchips anzusprechen. Bis XFree86 3.3.6 musste frau daher einen X-Server entsprechend des Chipsatzes (z.B. *XF86_Mach64* oder (den für die meisten Chipsätze geeigneten, aber recht rudimentären) *XF86_VGA16*) installiert haben, ab XFree86 4.0 (das gilt auch für die X.org-Implementation) gibt es nur noch ein *Binary* für alle Chipsätze; die hardwareabhängigen Teile sind in Module ausgelagert. Verwirrend wird es allerdings, wenn neben der 2D-Grafik 3D-Hardware-Unterstützung ins Spiel kommt: Dann muss nämlich auch der Kernel entsprechend gepatcht sein.
Die Konfiguration des X-Servers erfolgt in der Regel bei der Installation, kann aber immer auch nachträglich durchgeführt werden. Die meisten modernen Distributionen erkennen Grafikkarte und (wenn möglich) auch Monitor automatisch, allerdings schadet es nichts, diese Angaben, am besten die jeweiligen Handbücher zur Hardware, dabei liegen zu haben, um nach Fragen zu Horizontal- und Vertikalfrequenz (des Monitors), zum Chipsatz der Grafikkarte und zur Auflösung Auskunft geben zu können. Speziell für Nutzerinnen nichtkommerzieller Distributionen wie Debian ist das ein Muss!
Die Konfiguration landet beim XFree-X-Server in der Datei `XF86Config` (der Pfad kann je nach Distribution variieren, sollte laut *Filesystem Hierarchy Standard* aber `/etc/X11/XF86Config` sein). Einige Distributoren modifizieren den Namen leicht, wenn sie sowohl eine 3er als auch eine 4er Version anbieten (die neuere Version unterstützt einige Grafikkarten nicht mehr ). In der Syntax unterscheiden sich beide Konfigurationsdateien nämlich leicht. Beim X.org-Server lautet der Name der Konfigurationsdatei `xorg.conf`; auch sie gehört normalerweise ins Verzeichnis `/etc/X11`.
Später lässt sich die Datei mit einem Texteditor oder einem distributionsabhängigen Tool wie Yast (SuSE) bzw. DrakConf (Mandriva), `system-config-display` (Fedora Core) oder aber auch `XF86Setup` (XFree 3) und `xf86cfg` (XFree 4) vom XFree-Team bzw. mit `xorgcfg` ändern. Auskunft über das vorgesehene Tool gibt in der Regel das Handbuch der Distribution.

✎ Finde die X-Server-Konfigurationsdatei auf Deinem Rechner mit dem Kommando `locate`.

✎ Schau mit `more` in die Datei hinein. Du solltest darin weitere Aufgabengebiete des X-Servers finden, die wir uns gemeinsam anschauen. Suche mit dem `more`-Befehl `/` nach den Stichworten `FontPath`, `XkbLayout`, der `Section "Pointer"` (XFree 3) bzw. `Section "InputDevice"` (XFree 4/X.org)!

Wenn ein grafisches Programm, ein *X-Client*, sich auf einem Bildschirm darstellen will, fragt es bei einem beliebigen X-Server (es muss nicht der der lokalen Maschine sein) an, ob ihm das erlaubt sei. Darf die Benutzerin, mit deren Rechten der Client läuft, auf den X-Server zugreifen (weil es ohnehin ihr eigener X-Server ist oder weil sie oder ihr Rechner Zugriffsrechte auf einen entfernten X-Server hat), wird das grafische Programm auf dem jeweiligen X-Server dargestellt.





Noch recht jung ist das Debian-Women-Projekt, das sich zum Ziel gesetzt hat, etwas gegen die äußerst geringe Zahl im Debian-Projekt involvierten Frauen zu tun. Auch hier sind Männer willkommen, eine Einladung, die auf der Mailingliste (http://lists.debian.org/debian-women/) leider bereits öfter missbraucht wurde.

## 15.2 Deutschsprachige Portale

- http://www.linux.de/
- http://www.pro-linux.de/
- http://www.linux-community.de/
- http://www.linux-knowledge-portal.org/de/index.php
- http://www.go-linux.de/

## 15.3 Bücher und Zeitschriften

- *Das Linux-Anwenderinnen-Handbuch* (deutsch, vollständig online): http://www.linux-ag.de/linux/LHB/
- *Andamooka* (Kollektion frei verfügbarer Bücher – darunter auch einige nicht-englischsprachige – online, Site ist englisch): http://www.andamooka.org/
- *Open Books Project* (deutsche und englische O'Reilly-Bücher vollständig online): http://www.oreilly.de/openbook/
- *Monatszeitschrift LinuxUser* (ältere Ausgaben vollständig online, bei neueren Ausgaben ausgewählte Artikel) http://www.linux-user.de/
- *Monatszeitschrift EasyLinux* (aus demselben Verlag) http://www.easylinux.de/

## 15.4 Software-Portale

- http://freshmeat.net/
- http://linux.tucows.com/
- http://www.kde-apps.org/

## 15.5 Freie und Open-Source-Software (FOSS)

- *GNU* http://www.gnu.org/
- *Die Kathedrale und der Basar* http://www.linux-magazin.de/ausgabe/1997/08/Basar/basar.html

Shell bereitstellen. (Aktuelle Kernel umfassen mehrere Millionen Zeilen Quellcode, der Kernel 2.4.9 etwa bestand aus 3,7 Millionen Zeilen!)
Im Februar 1992 erschien die erste, am Manchester Computing Centre (MCC) von Owen LeBlanc erstellte Distribution als „MCC Interim release“. Im Dezember 1992 kommt mit Yggdrasil die erste Distribution auf CD heraus; die erste Slackware-Distribution und die erste, darauf basierende SuSE fallen ebenfalls in dieses Jahr. 1993 entstehen Red Hat und Debian. Wieviele unterschiedliche Distributionen es heute weltweit gibt, ist unbekannt, eine Schätzung aus dem Jahr 2001 sprach von über 188[37]. Diese Zahl dürfte mittlerweile weitaus höher liegen.
Seit April 1993 läuft das X-Window-System auf Linux, und am 16. April 1994 wird Versionsnummer 1.0 vergeben.
Im Oktober 1996 kommt die Idee zu KDE auf, die Version 1.0 erscheint 1998. 1999 folgt GNOME mit der Version 1.0. Spätestens zu diesem Zeitpunkt nimmt eine immer breitere Öffentlichkeit das Betriebssystem mit dem Pinguin wahr, der 1996 zum Maskottchen erkoren und „Tux“ getauft wurde.

## 15 Informationen und Downloads

Eine viel zu kurz geratene Zusammenstellung hilfreicher Adressen ...

### 15.1 Unter Frauen: Gedankenaustausch zu Linux und Co.

- Mailingliste *Lynn* (deutschsprachig):
  http://lists.answergirl.de/

  Die unter *lynn@lists.answergirl.de* erreichbare Linuxerinnenmailingliste bietet die Möglichkeit, sich unter Frauen über Linux auszutauschen. Einsteigerinnen sind ebenso gern gesehen wie „Profis“. Die Anmeldung erfolgt über die angegebene Webseite oder per Mail mit dem Subject: `subscribe lynn` an *ecartis@lists.answergirl.de*.

  Bitte sei so nett und stell Dich in einer ersten Mail an die Liste kurz vor. Das Listenarchiv unter http://lists.answergirl.de/lynn/ ist passwortgeschützt und steht allen Frauen offen, die mit mindestens einem Posting (Vorstellungsmail und/oder fachlicher Beitrag) auf der Liste sichtbar geworden sind.

  Als zusätzliches Kommunikationsangebot gibt es im IRCNet (z. B. irc.leo.org) den IRC-Kanal *#lynn-de*.

  Für Programmiererinnen interessant dürfte die Liste `ada` auf demselben Server sein. Die Mailinglisten sind eingebunden in das (deutschsprachige) *Technixen*-Projekt:
  http://www.technixen.net/

- *KDE-Women* (englisch):
  http://women.kde.org/

  Das Projekt, gedacht als Networking-Möglichkeit für KDE-Entwicklerinnen und -Nutzerinnen, leidet vor allem daran, dass es auf zu wenigen tragenden Schultern ruht. Der IRC-Kanal *#kde-women* im Freenode-IRC-Netz (z. B. irc.kde.org) und die Mailingliste (https://mail.kde.org/mailman/listinfo/kde-women) schließen männliche Beteiligung allerdings nicht aus.

- *Debian-Women* (englisch):
  http://women.alioth.debian.org/

[37] vgl. http://www.linux-community.de/Neues/story?storyid=1389





Auf diese Weise ist es sehr einfach, Unix-Maschinen nicht nur über die Kommandozeile, sondern auch grafisch aus der Ferne zu bedienen und zu warten; außerdem wird es möglich, Lasten zu verteilen: Die schmalbrüstige Workstation beschränkt sich darauf, die grafische Oberfläche eines Programms darzustellen, während die Berechnungen auf einem leistungsstärkeren Rechner ausgeführt werden.
Allerdings bringt diese Architektur auch Sicherheitsprobleme mit sich, die in „alten Unixtagen“ keine Rolle spielten. Frau sollte die Remote-Ausgabe grafischer Programme daher besser auf vertrauenswürdige lokale Netzwerke beschränken und für das Remote-Login auf jeden Fall *SecureShell* verwenden. (Das sogenannte *X-Forwarding* der SecureShell kann jedoch abgeschaltet werden, sodass es per Default nicht mehr möglich ist, die Darstellung grafischer, auf dem entfernten Rechner laufender Programme dem lokalen X-Server zu überlassen.)

✎ Logge Dich mit `ssh -X` *username@hostname* auf dem Rechner Deiner Nachbarin ein und starte dort das Programm `xeyes`. Auf welchem Bildschirm erscheint es? Warum?

### 2.5 Windowmanager

Um eines kümmert sich der X-Server jedoch herzlich wenig: Ob die verschiedenen X-Clients einheitliche Fensterrahmen und Buttons bekommen, mit denen sie über den Desktop bewegt, vergrößert oder verkleinert werden können, ist ihm ebenso egal wie die Frage, ob die Anwenderin alle Fenster schlecht zugänglich übereinander oder übersichtlich verteilt vorfindet.
Sich darum zu kümmern, ist die Aufgabe eines ganz speziellen X-Clients, des Windowmanagers. Erst dessen Einsatz beim Dirigieren der Client-Fenster macht sinnvolles Arbeiten auf der grafischen Oberfläche möglich. Durch Wahl des Windowmanagers entsprechend den eigenen Vorlieben und Bedürfnissen wird ein Unix-Desktop zur individuell angepassten Arbeitsoberfläche.
Die Auswahl reicht vom rudimentären EvilWM, der für minimale Fensterdekorationen und Tastaturbedienbarkeit steht, über den Klassiker `fvwm2` (funktional, stark konfigurierbar, aber für heutige „Sehgewohnheiten“ eher mit altbackenem Outfit) bis hin zum schrillen, durch *Themes* anpassbaren *Enlightenment*. Neue Projekte kommen hinzu, während die Weiterentwicklung so manchen früheren „Stars“ zeitweilig oder in Gänze einschläft.
Neben der beschriebenen Grundfunktionalität stellen viele Windowmanager Programmauswahl- und Aktionsmenüs bereit, die gern per Mausklick auf den Desktop-Hintergrund (Unix-Mäuse haben drei Maustasten!) hin erscheinen, *virtuelle Desktops* sorgen bei Vielfenster-Benutzerinnen für Übersicht. Manche Windowmanager bieten sogar ein gewisses Session-Management, das es erlaubt, den Windowmanager zu schließen und beim nächsten Start mit den zuletzt genutzten Anwendungen starten zu lassen. Natürlich unterscheiden sich die „Fensterherrscher“ auch nach der Art und dem Umfang ihrer Konfigurierbarkeit.

✎ Stell Deinen Kommilitoninnen ein bis zwei Windowmanager kurz (ggf. mit Bild) vor!

### 2.6 Desktop-Umgebungen

So individuell sich die Arbeitsoberfläche durch Auswahl und Konfiguration des Windowmanagers gestalten lässt, er behebt ein altes Problem des Unix-Desktops nicht: die Individualität der Anwendungen. Während Windows-Benutzerinnen in der Regel davon ausgehen können, dass sie links oben in der Menüleiste das *Datei*-, rechts das *Hilfe*-Menü erwischen und dass gängige Aktionen anwendungsübergreifend über ein und dasselbe Tastenkürzel erreichbar sind, hatten es Unix-Benutzerinnen hier weniger gut. Die Freiheit, sich nicht dem Style-Guide eines Herstellers beugen zu müssen, führte dazu, dass jede/r Programmierer/in seine/ihre GUIs nach eigenem

Gutdünken schrieb. So fehlt klassischen X-Clients nicht nur ein einheitliches Aussehen, sondern auch die Fähigkeit des *Drag&Drop* über Applikationsgrenzen.
Die einzige, zuverlässig funktionierende Kommunikationsmöglichkeit ist *Copy&Paste*. Dazu markiert frau den gewünschten Text mit der linken Maustaste und fügt ihn mit der mittleren in die Zielapplikation ein. Bei Zweitastenmäusen lässt sich der gemeinsame Klick auf beide Tasten zum „mittleren Mausklick“ umfunktionieren.
Dem beschriebenen Missstand begegnen *Desktop-Umgebungen* (auch *Desktop-Environments* genannt) wie KDE oder GNOME. Das sind Software-Suites, die zum einen einen (z. T. austauschbaren) Windowmanager, zum anderen konsistent gestaltete GUI-Applikationen mitbringen. Letzteres erreichen deren Entwickler/innen u. a. dadurch, dass Software, die für eine Desktop-Umgebung geschrieben wird, ein bestimmtes *GUI-Toolkit* benutzt: Qt für KDE und GTK für GNOME.
Dazu kommen „Hilfsprogramme“ wie Soundserver, Protokolle und Bibliotheken für die Kommunikation zwischen den Applikationen, Themes u. a. Auch wenn es zwischen Anhängern der einen und der anderen Umgebung oft fanatisch anmutende Grabenkämpfe gibt – die Entscheidung für oder gegen das eine oder das andere Environment sollte frau allein davon abhängig machen, was ihr persönlich besser zusagt.
Desktop-Umgebungen haben natürlich auch einen Nachteil: Die Ansprüche an die Hardware sind nicht zu vernachlässigen. Auf älteren Rechnern lässt sich mit einem schlanken Windowmanager oft produktiv grafisch arbeiten – die vielleicht moderner wirkende Oberfläche von KDE und GNOME zwingt ihn dagegen in die Knie.
Sofern die entsprechenden Bibliotheken installiert sind, lassen sich KDE-Applikationen selbstverständlich unter GNOME oder einem Standalone-Windowmanager starten und umgekehrt. Allerdings lernt ein guter alter X-Client nicht allein dadurch Drag&Drop, dass er plötzlich unter KDE laufen darf.

**Wie wähle ich meinen Windowmanager/meine Desktop-Umgebung?** Loggt frau sich grafisch ein, wird anschließend der Windowmanager/die Desktop-Umgebung gestartet, der/die in der Datei `~/.xsession` steht. Startet frau den X-Server mit dem Befehl `startx` „von Hand“, kommt die Datei `~/.xinitrc` zum Zuge. Steht in dieser Datei beispielsweise der Befehl `startkde`, startet KDE. Wichtig: Sollen sonstige Programme automatisch gestartet werden, müssen sie *vor* diesem Aufruf in die Datei eingefügt werden – handelt es sich um grafische Programme, darf frau zudem nicht vergessen, diese als Hintergrundprozess aufzurufen (siehe Abschnitt 8.3 auf Seite 45). Leider gibt es Distributionen, die sich an diese einfache Regel nicht halten oder die Sache durch undurchsichtigen Inhalt in besagten Dateien erschweren.
In den Menüs des *Display-Managers*, der bei grafischem Login Username und Passwort abfragt, aber auch in den „Startmenüs“ von Windowmanagern finden sich oft Einträge zum Wechseln in eine andere grafische Umgebung. Manche Distributionen wie Red Hat (und Fedora) bringen extra Tools (`switchdesk`) mit, die bei der Festlegung behilflich sein können.

✎ Überprüfe, ob der Poolrechner via Menü Alternativen bietet, und probiere sie durch!

✎ Für Mutige etwas später im Kurs (bitte nicht „zwischenrein“): Sichere ggf. die entsprechende Konfigurationsdatei und versuche, einen alternativen Windowmanager zu starten!



# Anhang

## 14 Geschichte

Am 25. August 1991 schickt der 21-jährige finnische Student Linus Torvalds folgendes Posting an die Minix[36]-Newsgroup:

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki


Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones.  This has been brewing
since april, and is starting to get ready.  I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
This implies that I'll get something practical within a few months, and
I'd like to know what features most people would want.  Any suggestions
are welcome, but I won't promise I'll implement them :-)

            Linus (torvalds@kruuna.helsinki.fi)

PS.  Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that's all I have :-(.
```

Das Projekt, für das er Mitstreiter/innen sucht, heißt zunächst *FREAX*, erst später bekommt es den Namen Linux.
Schon am 17. September steht die Version 0.01 auf einem ftp-Server zum Download bereit – 10.000 Zeilen Quelltexte. Dieser Kernel konnte von Diskette auf einem 386er booten und eine

[36] Das von Andrew S. Tanenbaum an der Vrije Universiteit Amsterdam zu Lehrzwecken entwickelte rudimentäre PC-Unix ist in seinem 1987 erschienenen Buch „Operating Systems, Design and Implementation“ kommentiert und komplett im Quellcode abgedruckt.

### 2.7 Clients, Server und andere Programme

Wie schon beim X-Server und den X-Clients zu sehen war, lässt sich die Client-Server-Architektur auf Unixbetriebssystemen nicht wegdenken. Auch heutige Linux-Distributionen folgen dieser Tradition weiterhin, und so gehört schon eine Menge Arbeit hinzu, um ein Linuxsystem so hinzubiegen, dass es weitgehend serverlos ist.
Die meisten Server werden beim Booten des Rechners gestartet und laufen dann von den Usern weitgehend unbeachtet im Hintergrund vor sich hin. Selbst wenn ein Client ihre Dienste anfordert, geht das in vielen Fällen von den Nutzerinnen unbemerkt vonstatten (auch wenn frau sich natürlich gezielt vergewissern kann, ob ein Dienst aktiv ist).
Solche Programme nennt frau *Daemonen* (von „Disk and Execution Monitor“); ihre Namen enden meist auf ein d. Sie verrichten die unterschiedlichsten Aufgaben: So sorgt der `kerneld` dafür, dass Kernelmodule geladen werden. Er tanzt jedoch insofern aus der Reihe, als dass es sich nicht um einen eigenständigen Prozess, sondern um eine Kernelfunktionalität handelt.
Der `syslogd` protokolliert („loggt“) die Ausgaben von Dienstprogrammen, sodass sie den Usern nicht auf die Nerven gehen, die Systemadministratorin aber bei Bedarf nachlesen kann, was passiert ist, der `crond` führt von den Nutzerinnen hinterlegte Aufgaben zu den angegebenen Zeiten aus...
Neben diesen „systemerhaltenden“ Daemonen lässt sich kaum ein Linuxsystem ohne Internetserver denken. Welche installiert sind und welche laufen, hängt jedoch von der Distribution und vom Installationsumfang ab.
Will frau dafür sorgen, dass sie sich remote (also von einem räumlich entfernten Rechner aus) auf einer Maschine einloggen kann, muss der SecureShell-Server `sshd` laufen. (Heutzutage, wo frau im Netz niemandem mehr trauen kann, sollte der `telnetd` zum unverschlüsselten Remote-Einloggen hingegen nicht mehr zum Einsatz kommen.)
Ein Mailserver ist eine gute Idee (und sei es nur, damit der Cron-Daemon (vgl. Seite 47) die Ausgaben der von ihm ausgeführten Programme an die lokalen User senden kann). Der Begriff „smtpd“ findet sich zwar hin und wieder, aber da Mail vermutlich der Internetdienst ist, bei der frau die größte Auswahl an Servern hat, hat er meist nur kategorisierende Bedeutung. Hier gibt es beispielsweise den sehr kryptischen, aber weit verbreiteten Sendmail, den eher exotischen Qmail, Postfix, Zmailer, smail, Exim ..., um die bekanntesten zu nennen. Allerdings spezialisieren sich die meisten Distributionen meist auf eine Software, sodass frau sich den Mailserver ihrer Wahl u. U. selbst besorgen muss.[8]
Diese „Mail Transfer Agents“ (MTAs) sind sowohl fürs Versenden als auch fürs Empfangen von Mail zuständig, aber da – anders als zu Jugendzeiten des Internets – heute lange nicht mehr jeder Internetrechner ständig und mit statischer IP-Adresse im Netz hängt, sorgen POP3- und IMAP-Server dafür, dass auch Menschen ohne eigenen MTA und DNS-Eintrag Mails *beziehen* können. Die entsprechenden Server gehören allerdings schon nicht mehr zur Standardausgabe von Distributionen, die sich auf den Desktop-Einsatz spezialisieren.
Als Webserver ist – selbst bei „Desktop-Distributionen“ – meist Apache dabei, oft läuft er von Haus aus, um Zugriff auf ein lokales Hilfesystem zu gewähren.

✎ Für Mail (smtp, „Simple Mail Transfer Protocol“) ist ein sogenannter *wellknown port* reserviert, hinter dem der Mailserver lauschen müsste, sofern einer läuft. Suche in der Datei `/etc/services` heraus, welche Portnummer das ist.

✎ Benutze das Kommando `telnet localhost` *portnummer*, um Dich mit dem SMTP-Port zu verbinden. Läuft ein Mailserver und wenn ja, welcher?

[8] Dass Qmail in der Regel fehlt, liegt allerdings nicht an den Distributoren, sondern an den eigenwilligen Lizenzbedingungen des Servers.

Auch Nameserver (Bind), Newsserver, IRC-Server, Listserver, FTP-Server ...– Server für jeden erdenklichen Internetdienst gehören zur Standardausstattung vieler Distributionen, mit der zunehmenden Ausrichtung auf Enduser allerdings nicht mehr bei allen.
Wichtig zu erwähnen wäre noch der „Superserver“ `inetd` oder `xinetd`. Sofern er läuft, müssen nicht alle anderen Netzdienste ständig laufen – er startet sie, wenn ein Client ihn entsprechend auffordert und die Konfiguration es ihm gestattet. Allerdings sollte frau aus Sicherheitsgründen auf ihn verzichten, wenn sie keinen der von ihm verwalteten Dienste – etwa Telnet oder FTP – bereitstellen will.
All diese Server müssen natürlich konfiguriert werden, nur für sehr gängige Szenarien bringen die Distributionen bereits entsprechende Voreinstellungen mit. Meist passiert das über im Verzeichnis `/etc` liegende ASCII-Konfigurationsdateien. Eine gewisse Vereinfachung bieten distributionsabhängige Konfigurationstools (wie `yast2` bei SuSE) oder distributionsunabhängige Werkzeuge wie (der aus sicherheitstechnischer Sicht recht bedenkliche) Webmin. Allerdings setzen auch diese Tools ein gewisses Verständnis des Dienstes voraus, den es zu konfigurieren gilt, und decken oft nur triviale Szenarien ab.
Wo es Server gibt, braucht frau auch Clients – sie gibt es mittlerweile für fast alle Dienste sowohl als GUI- als auch als Kommandozeilenprogramme, von denen jede auch nur annähernd auf den Desktop „schielende“ Distribution eine Auswahl mitbringt.

✎ Finde drei Mailprogramme (Mail User Agents, MUAs) für Linux und prüfe, ob sie auf Deinem Rechner installiert sind! Verwende dazu das Kommando `locate`!

Doch nicht alle Programme auf einem Linuxsystem lassen sich als Clients oder Server kategorisieren. Wer die Kommandozeile als mächtiges und schnelles Interaktionsmittel mit dem Betriebssystem kennengelernt hat, mag auf eine Reihe von Werkzeugen nicht mehr verzichten, die aus der Feder des GNU-Projekts (http://www.gnu.org/) stammen.

**Warum GNU?** Unix-Hacker und -Häcksen sind Spielkinder, die einem Sport besonders fröhnen: dem Spaß an *rekursiven Akronymen*, also selbstbezüglichen Abkürzungen. So steht GNU für „GNU is not UNIX“ und weist auf das ursprüngliche Ziel hin, ein freies unixoides Betriebssystem zu entwickeln, das nichts mit dem originalen AT&T Unix zu tun hat.
Die GNU-Leute (darunter der Begründer der Free Software Foundation (FSF), Richard Stallman) begannen mit den Tools, während der Betriebssystemkern des Projekts, GNU Hurd, bis heute noch nicht endusertauglich ist.
Als Linus Torvalds Anfang der Neunziger an einem Betriebssystemkern schrieb, dem die Tools fehlten, lag es nahe, beides zu vereinen. Aus diesem Grund sehen sich die FSF-Leute bei der Bezeichnung Linux für ein Linuxsystem unterrepräsentiert und bestehen auf der Bezeichnung GNU/Linux, wie sie im Namen der Debian-Distribution (Debian GNU/Linux) zu sehen ist.
Alle GNU-Utilities unterstehen (wie auch der Linux-Kernel und ein Großteil sonstiger Open-Source-Software) der GNU General Public Licence GPL, die im Wesentlichen besagt, dass der Quellcode der Software öffentlich zugänglich ist und frau daraus auch kopieren darf. Wenn dies geschieht, verlangt die Lizenz, dass der daraus abgeleitete Code ebenfalls der GPL unterstehen muss, was ihren *viralen Charakter* ausmacht.

Sie implementieren die klassischen Unix-Kommandozeilentools zum Suchen, Manipulieren und Anzeigen von Dateien und Prozessen und gehen in ihrer Funktionalität meist über die der jeweiligen Vorbild-Programme hinaus. Ob GNU-Version oder nicht: In klassischer Unixtradition





✎ Besitzt Du selbst eine Domain? Finde heraus, welcher Rechner die Mail für diese Domain annimmt. (Wenn Du selbst keine Domain hast, probiere `answergirl.de` aus!)

Interessiert eher, wer eigentlich hinter einer Domain steckt, gibt `whois` die Antwort, indem es in der Datenbank eines *Network Information Center* (NIC) nachschlägt. Um einen anderen als den als Default eingestellten Whois-Rechner zu fragen, kommt auch hier meist die Option `-h` zum Einsatz. Leider sind unter Linux zwei Versionen des `whois`-Programms im Umlauf, die sich unterschiedlich bedienen lassen – welche, sollte ein Versuch oder der Blick in die Manpage verraten:

```
whois -h whois.ripe.net informatica-feminale.de
whois informatica-feminale.de@whois.ripe.net
```

fragt `whois.ripe.net` nach den Registrierungsangaben für die Domain `informatica-feminale.de`. Allerdings gibt der Denic, der die .de-Domains verwaltet, aus Datenschutzgründen nur noch die Information heraus, ob eine Domain „connected“ ist oder nicht.

✎ Schlage die Registrierungsinformationen zu Deiner Domain oder der Deines Internetproviders nach!

✎ Welche Domain würdest Du gern registrieren? Schau nach, ob es sie schon gibt!

**Who is who?** Wer ist eigentlich sonst noch alles auf meinem Rechner eingeloggt? `who` gibt die Antwort. Und sollte nach einer durchwachten Nacht unklar sein, wer das eigentlich ist, die da vor dem Rechner sitzt, beantwortet der Rechner auch die Frage `who am i`.

✎ Was ist der Unterschied zwischen `who am i` und `whoami`?

Informationen über Accounts auf anderen Rechner gibt das `finger`-Kommando, das allerdings aus Datenschutzgründen immer öfter ohne Antwort auskommen muss.

✎ Prüfe, ob Du mit `finger UserID` und `finger UserID@remoterechner` Informationen zu Deinem Account lokal und auf einem anderen Poolrechner heraus finden kannst.

✎ „Fingere“ Deinen Uni-Account an! Bekommst Du eine Antwort?

✎ Log Dich per `ftp` oder `ncftp` (am besten zum Vergleich mit beiden) auf dem KDE-FTP-Server `ftp.kde.org` ein und lade zwei Dateien in einem Rutsch herunter.

### 13.2 Wer ist wo online?

Solange das Netz geht, ist alles in Ordnung. Wenn nicht, gibt es jede Menge Diagnosetools, mit denen frau den Problemen auf die Sprünge kommt. Einige gängige stellen wir kurz vor.

**Erreichbarkeit im Netz prüfen** Unsicher, ob frau überhaupt online ist oder ob der eigene Netzzugang funktioniert?

```
ping rechneradresse
```

spielt mit einem entfernten Rechner „Ping-Pong“, indem es ein Paket hinsendet und eins zurück haben will.

✎ Versuche, den Rechner Deiner Nachbarin anzupingen. Denke Dir einen Phantasierechnernamen aus und versuche, diesen zu erwischen. Achte auf die Unterschiede bei der von `ping` ausgegebenen Statistik!

**Den Weg zum Ziel finden** Welchen Weg (welche *Route*) nimmt ein Paket momentan zu einem Zielrechner? Hierauf gibt `traceroute` die Antwort. In der ausgegebenen Statistik werden auch Engpässe sichtbar.

✎ Lass Dir die Route zu einem Rechner Deiner heimischen Hochschule oder Deines privaten Internetproviders ausgeben!

**DNS-Informationen holen** Auch wenn wir uns mit dem „Domain Name System“ in diesem Kurs nicht beschäftigen können: Die Informationen, die ein Internetclient von einem *Nameserver* anfordert, um zu einer textuellen Rechneradresse die passende numerische IP-Adresse zu bekommen (oder umgekehrt, was sich dann *reverse lookup* nennt), oder den Mailserver herauszufinden, bei dem er die abzuschickende E-Mail abliefern soll, stehen auch den Usern zur Verfügung. So finden die Befehle

```
host -t mx informatica-feminale.de
dig mx informatica-feminale.de
```

heraus, welcher Rechner Mail für die Domain `informatica-feminale.de` entgegen nimmt (`mx` steht für „Mail Exchanger“).

```
host www.informatica-feminale.de
```

sucht die numerische IP-Adresse des Rechners `www.informatica-feminale.de`.

```
host -a informatica-feminale.de nameserver
dig @nameserver any informatica-feminale.de
```

finden alle DNS-Informationen über die Domain `informatica-feminale.de` heraus, fragen dazu allerdings nicht den Default-Nameserver, sondern *nameserver*.

✎ Finde die numerische IP-Adresse des Servers heraus, den Du normalerweise als Mailserver benutzt.





handelt es sich dabei jedoch immer noch um einfache, hochspezialisierte Werkzeuge, die sich bei Bedarf baukastenartig zu komplexen, ein bestimmtes Problem lösenden Kommandozeilen zusammenbauen lassen.
Dabei ist *einfach* wohl eher ein Understatement, da sich die meisten Werkzeuge durch eine oftmals sehr große Reihe von Optionen in ihrer Wirkung steuern lassen. Diese Optionen, auch *Flags* genannt, leitet in der Regel ein Minuszeichen „-“ ein, GNU- oder *long* Optionen beginnen mit zwei Minuszeichen, und manchmal darf oder soll frau sogar einfach nur einen Buchstaben hinschreiben.
Die Wirkung solcher Flags kann sich von der erwarteten unterscheiden, wenn frau mehrere Optionen miteinander kombiniert. Es sei zudem darauf hingewiesen, dass die möglichen Flags zu einem Kommando von Unix-Derivat zu Unix-Derivat differieren können. Gerade die GNU-Utilities kennen eine Reihe von Optionen, die ihre Pendants auf anderen Unixsystemen nicht beherrschen.
Während Flags meist optional sind, benötigen die meisten Kommandos Argumente (zu den Ausnahmen zählen sehr einfache Befehle wie `pwd` („print working directory“) oder `exit`). So kann frau z. B. `mkdir` zum Erstellen eines Verzeichnisses nicht ohne Angabe eines Verzeichnisnamens aufrufen.

✎ Was tun `pwd` und `exit`?

✎ Was passiert, wenn frau `mkdir` ohne Argument aufruft? Warum?

## 3 Hilfe zur Selbsthilfe

Selbst wenn sich viele Befehlsnamen anhand englischer Begriffe gut merken lassen, überfordern die gut 1500–2500 auf der Kommandozeile aufrufbaren Programme (je nach Installationsumfang sogar mehr)[9] auch das Hirn erfahrener Nutzerinnen. Deshalb gilt auch unter Linux: Es kommt nicht so sehr darauf an, wieviel frau auswendig weiß – sie sollte nur wissen, wo sie im Zweifelsfall Hilfe findet.
Die bietet bereits die Shell durch clevere Bedienmöglichkeiten. So zeigt die `bash` (vgl. Kapitel 2.2, Seite 9) auf doppeltes Drücken der *Tab*-Taste hin alle Kommandos an, die im Suchpfad enthalten sind.[10] Ein e x Tab wiederum entlockt der Shell alle Befehle aus dem Suchpfad, die mit `ex` beginnen. So kann sich frau an den Namen eines gesuchten Befehls herantasten – sobald sie die Bash mit sovielen Buchstaben gefüttert hat, dass die eine eindeutige Antwort geben kann, ergänzt die Shell die richtigen Buchstaben von sich aus.
Diese *Tab-Komplettierung* funktioniert aber leider nicht für Kommandoflags und erst recht nicht bei Spitzfindigkeiten in der Bedienung eines Befehls. Ohne griffbereite Dokumentation geht es da nicht, und die sollte möglichst direkt auf dem Rechner zugänglich sein.

### 3.1 Selbstauskunft

Ganz allgemein kann frau in den Verzeichnissen `/usr/doc` und (oft auch) `/usr/share/doc` auf der Suche nach (in der Regel englischsprachiger) Dokumentation zu einzelnen Programmpaketen fündig werden. Was da liegt, variiert von Rechner zu Rechner.
Auf `.gz` (vgl. Seite 61) endende, komprimierte (Text-)Dateien lassen sich auf vielen Linux-Systemen mit `zless` anschauen, ohne sie explizit entpacken zu müssen.

[9] Etwa 400 davon finden sich grundsätzlich auf den meisten Unixsystemen.
[10] Der Suchpfad sind die Verzeichnisse, in denen die Shell nach ausführbaren Dateien sucht. Programme in diesen Ordnern lassen sich einfach durch ihren Dateinamen – ohne Angabe eines Verzeichnisses – aufrufen.

**man** Eine sichere Bank, was Informationen zu Kommandozeilenbefehlen, C-Funktionen und Konfigurationsdateien anbelangt, sind die sogenannten *Manpages* (Kurzform für „manual pages“). Diese Online-Hilfeseiten, die es auf fast jedem System gibt, erklären in einer leider meist sehr abstrakten Form die Syntax und Funktionalität einer Datei oder Funktion. Viele liegen auch in deutscher Sprache vor, aber nicht alle Distributionen liefern sie mit. Im Zweifelsfall ist ohnehin die englische Version maßgeblich, sodass gerade erfahrene Benutzerinnen lieber gleich das Original zur Hand nehmen.
Zum Ansehen der in `man`-Unterverzeichnissen (z. B. `/usr/man` oder `/usr/local/man`) sortiert abgelegten ASCII-Dateien im *roff*-Format nutzt frau den Befehl `man`:

```
man fstab
man man
man -k password
```

Die Anzeige übernimmt ein sogenanntes *Pager-Programm*, zum Beispiel das Kommando `more`, das wir bereits in Übungen verwendet haben. Unter Linux kommt allerdings meist sein mächtigeres Pendant `less` zum Einsatz, das sich beim Erreichen der letzten anzuzeigenden Zeile nicht selbst beendet, sondern mit (q) („quit“) dazu überredet werden will.
Typisch für Manpages ist ihre Aufteilung in Abschnitte wie *NAME*, *SYNOPSIS* oder *DESCRIPTION*, um einige zu nennen. Der zweite enthält eine Kurzfassung aller möglichen Parameter in (nicht immer strikt verwendeter) *Backus-Naur*-Schreibweise. Eckige Klammern markieren dabei Parameter und Optionen, die frau je nach Bedarf weglassen darf.

✎ Was tut `man -k`? Wofür könnte das k stehen?

✎ Was sind *cat pages*? Nutze die Manpage zu `man`!

Oben links und rechts in einer Manpage steht nicht nur das Stichwort, zu dem die Hilfe gehört, sondern auch eine Zahl in runden Klammern. Damit hat es folgende Bewandtnis: Wenn ein Kommandozeilenbefehl *und* eine gleichnamige Datei existieren, soll es auch zwei getrennte Manpages geben. Dazu wird jede Manpage einer *Sektion* zugeordnet: Nr. 1 enthält Hilfen zu Benutzerkommandos, während Nr. 5 das Format der gleichnamigen Datei beschreibt. Die Sektionsziffer taucht immer wieder auf, z. B. auch im Namen des Verzeichnisses, in dem die unformatierte Manpage liegt.

✎ `man man` bzw. `man 1 man` beschreibt nicht, welche Sektionen es gibt. Finde mit `man -k man` oder `apropos man` heraus, in welchen anderen Sektionen `man` auftaucht und lies nach, welche Sektionen es gibt.

Auch wenn das Benutzen der Manpages anfangs gewöhnungsbedürftig ist, geht dies mit zunehmender Erfahrung leichter. Ärgerlich ist, dass speziell in den letzten Jahren entstandene GUI-Programme selten eine Manpage mitbringen. Insofern sind sie eher ein Nachschlagewerk für die Kommandozeile.

**info** Manpages haben auch Nachteile – sie lassen sich zum Beispiel nicht verlinken. Das GNU-Projekt hat deshalb ein eigenes Onlinehilfe-System namens *info* geschrieben. Ruft frau das gleichnamige Kommando ohne weitere Parameter auf, bekommt sie eine Übersicht über alle auf dem System installierten Info-Seiten.
Mit einem Sternchen `*` ein- und zwei Doppelpunkten ausgeleitete Einträge (wie `* Help-Small-Screen::` in der Info-Seite zu `info`) kann frau mit den Pfeiltasten anwählen und mit (↵) auf die darin verlinkte Seite springen. Damit und mit den Buchstabenkommandos (p) („previous page“),





```
ssh -l UserID rechneradresse
ssh UserID@rechneradresse
```

`ssh` steht dabei für *SecureShell*, und der Name sagt bereits, warum sie `telnet` zum Einloggen auf anderen Rechner vorzuziehen ist: Beim Anmelden auf Remoterechnern via `telnet` o. ä. werden sämtliche Daten (einschließlich des Passworts!) im Klartext übertragen, d. h., sie können von Unbefugten abgehört werden. Dem beugt `ssh` vor, indem der gesamte Datenverkehr verschlüsselt wird. Netter Nebeneffekt: Frau kann (sofern sie lokal unter X arbeitet) auf dem Remoterechner Programme mit grafischer Oberfläche starten und bekommt sie auf dem lokalen Bildschirm angezeigt. Allerdings bieten Administrator/innen auch diese Eigenschaft zunehmend seltener an, denn dank Problemen mit X können sich hierbei Sicherheitslücken auftun.
Will frau sich auf ihrem Rechner per SecureShell einloggen, muss sie den SecureShell-Daemon `sshd` installieren und starten. Seine Konfiguration erfolgt über die ASCII-Datei `/etc/ssh/sshd_config`, während in `/etc/ssh/ssh_config` die globale Konfiguration des Clients vorgenommen wird.
Über `ssh` ist es auch möglich, andere Protokolle zu *tunneln*, sodass beispielsweise ein sicherer Zugriff auf einen POP-Server möglich wird.

**scp** Ebenfalls im SecureShell-(Client-)Paket enthalten ist `scp`, mit dem frau auf sicherem Weg Daten von Rechner zu Rechner kopieren kann:

```
scp lokale_datei UserID@rechneradresse:/tmp/
```

kopiert die Datei *lokale_datei* im aktuellen Verzeichnis ins `/tmp`-Verzeichnis auf dem Remoterechner. Die UserID auf dem entfernten Rechner sowie das `@` kann frau weglassen, wenn sie auf dem lokalen Rechner unter demselben Namen eingeloggt ist. Mit der Option `-r` („rekursiv“) lassen sich komplette Dateibäume fernkopieren:

```
scp -r UserID@rechneradresse:~/programme .
```

dupliziert das Verzeichnis `programme` aus dem Homeverzeichnis von *UserID* auf dem entfernten Rechner im aktuellen Verzeichnis auf der lokalen Maschine.

✎ Bitte Deine Nachbarin, ein Verzeichnis in ihrem Homeverzeichnis mit Rechten zu versehen, die es Dir erlauben, es von ihrer Maschine auf Deine zu kopieren. Logge Dich auf Ihrem Rechner ein und schau nach, ob sie dies getan hat. Kopiere das Verzeichnis auf Deinen Rechner.

**ftp** Bevor es `scp` gab, wurde das „File Transfer Protocol“ `ftp` zum „Fernkopieren“ von Daten benutzt. Dieser Einsatzzweck verliert an Bedeutung, da FTP ebenfalls alles im Klartext überträgt. Weiterhin populär ist das Protokoll allerdings als *Anonymous FTP* zum Herunterladen von Software u. ä. von FTP-Servern.
Als Username gilt für diese Lese-Zugriffe auf die Dateihierarchie des FTP-Servers zumeist `ftp` oder `anonymous`, als Passwort die eigene E-Mailadresse. Beides schicken (grafische) FTP-Clients oder Webbrowser gern von sich aus an den Server.
Will frau hingegen mit dem klassischen Kommandozeilenclienten `ftp` auf FTP-Server zugreifen, muss sie beides von Hand eingeben. Bequemer ist hier der oft installierte FTP-Client `ncftp`.
Im FTP-Clienten gibt frau FTP-Befehle ein, die meist auch auf das Kommando `help` hin aufgelistet werden. So wechselt `cd` das Verzeichnis auf dem Server, `bin` schaltet in den binären Übertragungsmodus, der dafür sorgt, dass auch Nicht-Textdateien heil bei der Empfängerin ankommen, `lcd` wechselt das Verzeichnis auf dem lokalen Rechner, `ls` oder `dir` listet den Verzeichnisinhalt des aktuellen Verzeichnisses auf dem Server auf, `get` lädt die als Argument angegebene Datei herunter und `mget` mehrere, auch durch Wildcards spezifizierte.

```
tar -xf archiv.tar
```

entpacken.
Die unter Linux verwendete GNU-Version von `tar` kann nach dem Ein- oder vor dem Auspacken einen (De-)Kompressionsschritt einfügen, um die Archivdatei klein zu halten. Dazu greift das Programm wahlweise auf die externen Komprimierungstools `gzip` oder (nur bei neueren Versionen von GNU-`tar`) `bzip2` zurück. So gibt

```
tar -tjf archiv.tar.bz2
```

(bei älteren Versionen `tar -tIf` den Inhalt eines mit `bzip2` gepackten Tarballs aus, während

```
tar -czvf archiv.tar.gz .*
```

das entstehende Tar-Archiv mit `gzip` packt. Die Option `-v` („verbose“) sorgt dafür, dass `tar` während der Arbeit ein paar mehr Informationen rausrückt.

✎ Welche Dateien packt `tar -czvf archiv.tar.gz .*` eigentlich ins Archiv?

```
tar --help|less
```

spuckt eine Kurzfassung aller `tar`-Optionen aus, wobei `less` dafür sorgt, dass sie seitenweise lesbar sind. Die `--help`-Option funktioniert übrigens bei den meisten Kommandozeilentools, auch wenn sie nicht vorgesehen ist: Im Fall einer Fehlbedienung geben die meisten Programme nämlich von sich aus genau diese Kurzhilfe aus.

✎ Pack alle Dateien je zweier Unterverzeichnisse Deines Homeverzeichnisse in ein `tgz`- und ein `tar.bz2`-Archiv. Prüfe die Archive, und pack sie in `/tmp` wieder aus. Pack die Archive anschließend in zwei Schritten in einem neu angelegten Verzeichnis aus: Erst dekomprimieren mit `gunzip` bzw. `bunzip2`, dann auspacken mit `tar`.

# 13 Netzwerk

Die Entwicklung des Internets und der Unixbetriebssystemfamilie hängt eng zusammen, und die Entwicklung von Linux ist ohne Netz ebenfalls nicht denkbar. So verwundert es nicht, dass Linux von vornherein mit allem ausgestattet ist, um als Client und/oder Server Teil des Internets zu sein. In diesem Abschnitt beschäftigen wir uns allerdings nicht mit den Grundlagen wie dem *TCP-IP-Stack*, die im Kernel implementiert sind und konfiguriert werden, sondern nur mit einigen wenigen Internet-Client-Programmen und Diagnosetools für die Kommandozeile.
Das Aufsetzen von Mail-, Web-, News- u.a. Internetservern könnte uns ebenso wie Netzwerkgrundlagen und das Konfigurieren des Internetanschlusses in mehreren Zusatzkursen beschäftigen, sodass wir es hier ebenfalls nicht ansprechen können.

## 13.1 Im Netzwerk bewegen

**ssh** Ähnlich wie das Unixurgestein `telnet` erlaubt es das Kommando

```
ssh rechneradresse
```

sich auf einem entfernten Rechner einzuloggen und dort so zu arbeiten, als säße frau lokal daran. Wenn sie auf dem Remoterechner eine andere UserID hat, gibt sie diese mit der Option `-l` oder wie in einer E-Mailadresse vor dem @-Zeichen an:





(n) („next page“), (u) („page up“), (d) („page down“) und (q) („quit“) kommt frau schon sehr weit.

✎ Vergleiche Man- und Infopage zum Kommando `groups`!

# 4 Dateisystem und Dateibaum

## 4.1 Dateisysteme

Egal, ob es sich nun um Programme, Dokumentations- oder sonstige Dateien handelt – sie müssen in jedem Fall so auf der Festplatte abgelegt werden, dass Programme wie die Shell auf sie zugreifen können. Diese Aufgabe hat ein Dateisystem, was letzten Endes nichts anderes ist als ein Kernel-„Treiber“, der sich ums strukturierte Ablegen der Daten auf einem Massenspeicher wie Festplatte, Diskette oder was auch immer kümmert und sie auf möglichst effiziente Weise darauf wiederfindet.
Bevor eine solche, betriebssystemspezifische Datenverwaltungsstruktur aufgebracht wird, werden Massenspeicher zunächst partitioniert, also in eine oder mehrere logische Verwaltungseinheiten, die Partitionen, „zerteilt“.
Auf einer Partition finden sowohl Verwaltungs- als auch Nutzdaten ihren Platz. Sinnvoll ist es zum Beispiel, die persönlichen Daten der Benutzerinnen (die „Home-Verzeichnisse“) auf einer anderen Partition abzulegen als die Systemdaten. Von entscheidender Bedeutung beim Hochfahren des Systems ist die sogenannte Root-Partition, deren Daten bei einem gebooteten System immer unterhalb des allerobersten „Wurzel-Verzeichnisses“ / liegen.
Eine Partition stellt sich dem Betriebssystem als eine lineare Kette von meist 1 kB, also 1024 Bit, großen Speicherblöcken dar.[11] Die Dateisysteme organisieren diese Blöcke so, dass sie darauf ihre Daten finden können. Außer bei sehr kleinen Dateien legen sie dabei die Verwaltungsinformationen zum Auffinden der Daten (die *Inodes*, kurz für „Information Nodes“) und die Daten selbst getrennt ab.
Die *Inodes* enthalten den Dateinamen, die Dateiattribute, die Größe und, falls die Datei größer ist als ein Datenblock, Zeiger auf eine Anzahl Datenblöcke, in denen die Daten der Datei liegen. Diese Blöcke können bei Bedarf mit weiteren Zeigern auf zusätzliche, von der Datei beanspruchte Blöcke verweisen.
Linux unterstützt eine ganze Reihe von Dateisystemen, die sich in ihren Interna deutlich voneinander unterscheiden. Die gute Nachricht: Für normale Nutzerinnen spielen diese Feinheiten (zumindest, was die Linux-eigenen Dateisysteme betrifft), kaum eine Rolle. Eine Auswahl:

| | |
|---|---|
| minix | Minix (kaum noch verbreitet, und wenn, dann für Disketten) |
| ext2 | das langjährige Standard-Dateisystem von Linux, das diese Stellung langsam einbüßt |
| ext3 | Nachfolger von Ext2, unterstützt Journaling; neben ReiserFS am weitesten verbreitet und die sinnvollste Lösung für Normalbenutzerinnen |
| ReiserFS | modernes Journaling-Dateisystem, wird speziell von SuSE gepusht |
| JFS | IBMs Open-Source-Beitrag zur Vielfalt der Journaling-Dateisysteme im Linux-Kernel |
| XFS | SGIs Open-Source-Beitrag zur Vielfalt der Journaling-Dateisysteme im Linux-Kernel |
| iso9660 | CD-ROM |

*Fortsetzung folgt ...*

[11] Bei den meisten Linux-Dateisystemen lässt sich die Blockgröße beim Partitionieren festlegen. Eine Ausnahme macht ReiserFS, das den Vorteil variabler Blockgrößen hat.

*... Fortsetzung*

| | |
|---|---|
| msdos | DOS |
| vfat | Windows 9x |
| nfs | „Network File System“, auf Unix-Systemen die verbreitetste Variante, auf auf entfernten Rechnern liegende Datenträger so zuzugreifen, als handle es sich um einen lokalen Datenspeicher |
| ntfs | Windows NT; stabil ist unter Linux bislang nur der Lesezugriff auf NTFS-Partitionen möglich |
| swap | Swap-Partitionen oder -Dateien (Auslagerungsbereiche, wenn's im RAM zu knapp wird) |
| proc | virtuelles Filesystem des Linux-Kernels zum Zwecke der Prozessverwaltung, das nur zur Laufzeit existiert |

**Minix** Nur noch aus historischen Gründen interessant ist das allererste Linux-Dateisystem, das vom Betriebssystem *Minix* stammt. Dieser UNIX®-Version-7-Nachbau von Andrew S. Tanenbaum, einem Professor an der Freien Universität Amsterdam (siehe auch Seite 67), inspirierte Linus Torvalds dazu, sein eigenes Betriebssystem zu schreiben. Offiziell unterstützen auch heute noch die meisten Distributionen das Minix-Dateisystem, das nur eine Partitionsgröße von 64 MB erlaubt und daher bestenfalls für Disketten etc. sinnvoll ist.

**Moderne Linux-Dateisysteme** Über viele Jahre hinweg war das vom Linux-Kernel-Team entwickelte *ext2* („second extended filesystem“) mehr oder weniger ein Synonym für „Linux-Dateisystem“. Diese unangefochtene Dominanz endete, als Firmen wie IBM und SGI auf den Linux-Zug aufsprangen und speziell für unternehmenskritische Einsätze konzipierte Dateisysteme nach Linux portierten und als Open-Source-Software freigaben. Genau wie JFS, XFS und das von Hans Reiser entwickelte ReiserFS gehören diese mittlerweile zur Standardausstattung des Linux-Kernels und sorgen für die Qual der Wahl.
Von diesen Entwicklungen getrieben, machten sich auch die Ext2-Entwickler Gedanken, wie sich das äußerst robuste Arbeitspferd Ext2 mit moderneren Eigenschaften wie Journaling ausstatten ließe. Das Ergebnis *ext3* ist seit der 2.4er Kernelserie mit „an Bord“. Im Gegensatz zu den übrigen Dateisystemen ist es zu Ext2 abwärtskompatibel. Datenverluste oder Boot-Probleme[12], wie sie bei „der Konkurrenz“ gelegentlich auftreten, kommen nicht vor.
In der Tat spielen die Alternativen ihre Vorteile (z. B. bei der Geschwindigkeit) meist nur bei speziellen Einsatzzwecken aus, die für „Normalbenutzerinnen“ nicht relevant sind. Insofern gibt es keinen Grund, sich hier die Wahl schwerer als nötig zu machen. Aber auch wenn der Distributor Ext2 als Standard-Filesystem vorsieht, ist das kein Anlass zur Panik, solange frau auf Journaling verzichten kann.

**Journaling Filesystem:** Stürzt der Rechner während einer Schreiboperation ab, kommt es zu Inkonsistenzen im Dateisystem. Verwaltungs- und Dateneinträge stimmen nicht mehr überein – die Einträge müssen geprüft und bereinigt werden.
Bei großen Platten dauert diese Überprüfung sehr lange, was vor allem bei Server-Systemen stört. Daher gibt es sogenannte *Journaling Filesysteme*. Diese schreiben sämtliche Dateioperationen „in einem Journal“ mit. Stürzt das System ab, müssen nur die Operationen nachgeprüft werden, die noch nicht beendet waren, was natürlich viel schneller geht.
Journale sind allerdings keinesfalls ein Ersatz für Backups...

[12] ... wobei an manchen Pannen nicht die Dateisystementwickler, sondern ein nachlässiger Distributor schuld ist...





**ASCII-Dateien schöner ausdrucken** Um Textdateien papiersparend auszudrucken, kann frau auf die Filter `a2ps` oder `enscript` (je nachdem, was installiert ist oder besser gefällt) zurückgreifen:

```
enscript -2 -M A4 textdatei
```

gibt *textdatei* zweispaltig im Hochformat auf A4-Papier (`-M` für „Medium“) auf dem Default-Drucker aus. Zusätzlich wird auf jeder Seite eine Headerzeile mit dem Namen der Datei, dem Ausdrucksdatum und der Seitennummer gedruckt.

```
a2ps -2 -M A4 textdatei
```

hingegen druckt zwei logische Seiten im Querformat A4 nebeneinander auf ein Blatt. Die logischen Seiten werden umrandet und mit hübschen Headern versehen. Zudem sorgt `a2ps` automatisch für Syntaxhighlighting, wenn es sich um Texte in einer gängigen Programmier- oder Auszeichnungssprache handelt.
Das kann `enscript` mit der Option `-E` ebenfalls:

```
enscript -2 -M A4 -E -o textdatei.ps textdatei
```

Die `-o`-Option („output“) funktioniert auch bei `a2ps` und sorgt dafür, dass das Ergebnis nicht auf dem Defaultdrucker (bzw. dem hinter `-P` angegebenen Drucker) heraus kommt, sondern in einer PostScript-Datei abgelegt wird. Letztere kann frau mit PostScript-Viewern wie `gv`, `kghostview` oder `ghostview` anschauen.

✎ Drucke eine Textdatei so in eine Datei, dass vier logische Seiten auf einer Druckseite zu liegen kommen. Schau sie mit einem PostScript-Viewer an!

**Papier sparen mit mpage** Auf vielen Systemen ist `mpage` installiert, das mehrere logische Seiten sowohl aus PostScript- als auch aus Textdateien ähnlich `a2ps` auf einer Druckseite ausgibt, sie bei Bedarf mit Rahmen umgibt und mit Header- und Fußzeilen versieht. Auch `mpage` kennt die Optionen `-o` und `-P`. Am Inhalt der auszudruckenden Seiten nimmt dieses Tool allerdings anders als `a2ps` und `enscript` keine Veränderungen vor.

**Packen und Entpacken** Beim Installieren von Software haben wir bereits mit dem „Tape Archiver“ `tar` Bekanntschaft geschlossen (vgl. Seite 44). Dieses Tool hat seinen Namen daher, dass es ursprünglich dazu entwickelt wurde, um Backups auf Bändern zu machen. Das geht auch heute noch, doch viel öfter wird `tar` dazu verwendet, um mehrere Dateien in einer (Archiv-)Datei zusammenzufassen. So packt

```
tar -cf archiv.tar .*
```

alle Punktdateien im aktuellen Verzeichnis zum Archiv *archiv.tar* zusammen (`-c` für „create“). Die Option `-f` muss unbedingt direkt vor dem Namen der geplanten Archivdatei stehen, denn sie entscheidet, dass `tar` nicht nach einem Bandlaufwerk sucht und stattdessen in eine Datei schreibt.
Das entstandene Archiv lässt sich mit

```
tar -tf archiv.tar
```

überprüfen (bei unbekannten *Tarballs* ein wichtiger Schritt, um zu sehen, wohin die darin enthaltenen Dateien beim Entpacken geschrieben werden) und mit

## 12 Nützliches Kleinzeug

**Taschenrechner** Natürlich gibt es auch für Linux Unmengen an grafischen Taschenrechnerprogrammen, die sich wie `kcalc` oder `xcalc` auch weitgehend mit der Tastatur bedienen lassen.
Doch es geht natürlich auch vollkommen auf der Kommandozeile. Neben `calc` (das mittlerweile eher Seltenheitswert in den Standardinstallationen besitzt), heißt das Standardprogramm hier `bc`. Dies ist keineswegs nur ein Taschenrechner, sondern eine Programmiersprache für sich. Wir beschränken uns in dieser Vorstellung allerdings auf den interaktiven Betrieb.
Nach dem Aufruf von `bc` rückt der Cursor auf die erste leere Zeile und wartet auf Eingaben. ⏎ gibt den Befehl zum Rechnen:

```
6.9+4.5
11.4
```

Dabei ist zu beachten, dass die Division per Default ganzzahlig berechnet wird:

```
6/4
1
6%4
2
```

Das lässt sich ändern, indem frau die Anzahl der auszugebenden Nachkommastellen mit der Spezialvariablen `scale` (der Default ist scale=0) festlegt:

```
scale=3
3.11/2
1.555
scale=1
3.11/2
1.5
3.11%2
.11
```

Die Modulo-Operation `%` gibt dabei jeweils den Rest aus, der bei der entsprechenden Genauigkeit übrig bleibt.
Mit den Pfeiltasten kann frau ältere „Befehle“ recyclen. Das jeweils letzte Ergebnis wird in der Variablen `last` gespeichert und kann durch Angabe dieses Variablennamens weiterverwendet werden:

```
8*4
32
last-30
2
```

Die Zahlensysteme für Ein- und Ausgabe legt frau für die aktuelle Session mit den Variablen `ibase` und `obase` fest. So rechnet der folgende Dialog die Zahl FF im hexadezimalen Zahlensystem ins oktale um:

```
ibase=16
obase=8
FF
377
```

Für komplexere Berechnungen sei die Manpage ans Herz gelegt.

✎ Übe das Rechnen mit `bc`!





**Ext2 und Ext3** Aus Anwenderinnensicht wichtig sind folgende Ext2/3-Eigenschaften:

- Dateinamen können bis zu 255 Zeichen lang sein.
- Groß- und Kleinschreibung werden unterschieden.
- Alle Zeichen – bis auf den Slash / und Nullbytes – sind in Dateinamen erlaubt. (Allerdings erleichtert frau sich die Arbeit, wenn sie auf Sonderzeichen verzichtet, da viele davon in der Shell eine eigene Bedeutung haben.) Es ist sogar möglich, einen aus Leerzeichen bestehenden Dateinamen zu bilden.

Das Ext2-Dateisystem bietet noch einen besonderen Service: Beim Mounten wird im Superblock der Partition ein sogenanntes Valid-Flag auf 0, und beim Unmounten wieder auf 1 zurückgesetzt. Beim Booten wird dieses Flag überprüft, und das Dateisystem gegebenenfalls mit dem Befehl `fsck` („file system check“) geprüft. Außerdem erhöht sich bei jedem Mounten ein Zähler im Superblock, der es ermöglicht, nach einer bestimmten, durch `tune2fs` einstellbaren Anzahl von Mounts und/oder Tagen das Dateisystem zu überprüfen, auch wenn das Valid-Flag korrekt gesetzt ist.
Beide Informationen wertet der Kernel beim Booten aus. Sie führen dann gegebenenfalls zu den Meldungen wie

```
/dev/hda1: was not cleanly unmounted, check forced
/dev/hda1 has reached maximal mount count, check forced
/dev/hda1 has gone too long without beeing checked, check forced
```

Bei Ext3 ist dieser Check nicht mehr nötig; hier kommt dem Kommando `tune2fs` eine ganz besondere Rolle zu: Mit der Option `-j` lässt sich eine Ext2-Partition zu Ext3 machen, ganz einfach, indem frau das Journal zuschaltet. Möchte frau wieder zu Ext2 zurück, schaltet sie es mit `tune2fs -O ^has_journal` *Partition* wieder ab. Beide Operationen sollte frau jeweils im Single-User-Wartungsmodus (siehe Kapitel 9 ab Seite 48) ausführen.
Das ist übrigens auch der Modus, in den das System rebootet, wenn die Schäden so groß sind, dass `fsck` mit seinem Reparatur-Latein am Ende ist. Mit `/sbin/fsck` *Partition* kann `root` dann eine manuelle Reparatur anstoßen. `fsck` stellt dann viele Fragen, die frau am besten mit (y) beantwortet. Kann das Programm Datenfragmente partout keiner Datei mehr zuordnen, landen diese im Verzeichnis `lost+found` der jeweiligen Partition. Die Systemadministatorin kann die dort abgelegten Dateischnipsel dann durchschauen und so ggf. der Erinnerung nach zumindest ein paar Textdateien reparieren. Wenn alles schief geht, bleibt die Hoffnung auf ein funktionierendes Backup...

**Reiser-Dateisystem** Ebenfalls seit Kernel 2.4 gehört das Reiser-Dateisystem *ReiserFS* zur Ausstattung des Linux-Kernels. Dieses Journaling Filesystem zeichnet sich dadurch aus, dass es die Daten in einem binären Baum verwaltet, was vor allem bei vielen kleinen Dateien einen enormen Performance-Gewinn nach sich zieht, aber auch den Plattenplatz optimal ausnutzt. Allerdings gibt es immer mal wieder Berichte darüber, dass ReiserFS es mit der Datensicherheit nach Abstürzen nicht ganz so genau nimmt.

**Virtuelles Dateisystem proc** Außer diesen Dateisystemen, die frau bei der Installation auf einen Datenträger aufbringt, gibt es welche, die nur der Linux-Kernel selbst zur Laufzeit erzeugt, um den Zugriff auf Systemparameter des laufenden Systems zu ermöglichen. Diese Daten liegen nicht auf separaten Partitionen, sondern im Arbeitsspeicher, und werden sichtbar, sobald frau auf sie zugreift.
Bei *proc* handelt es sich um ein solches virtuelles Dateisystem, in dem z. B. Informationen über laufende Prozesse und erkannte Hardware abgerufen oder Kernel-Parameter verändert werden können.
Es enthält keine realen Dateien, die auf der Festplatte Speicher wegnehmen, daher ist es auch nicht schlimm, wenn die Datei `/etc/kcore` riesig ist. Ihre Größe zeigt lediglich die Größe des Arbeitsspeichers an, sie selbst beansprucht aber keinen Platz. `root` darf den Inhalt des Arbeitsspeichers in der Regel auch lesen, es sei denn, es kommen – wie z. B. SELinux[13] bei Fedora Core – spezielle Schutzmechanismen zum Einsatz.

✎ Wie groß ist der Arbeitsspeicher Deines Systems?

✎ Schau zuhause in Deinen Arbeitsspeicher rein. Aber Vorsicht, *Pager* wie `more` oder `less` bekommen Probleme mit den nicht druckbaren Zeichen, also lieber vorher mit `strings` filtern:
`cat /proc/kcore | strings | less`

## 4.2 Mounten

Jede Partition, die Linux verwenden soll, muss gemountet, also in den Dateibaum eingehängt, werden, damit ein Zugriff auf die darauf befindlichen Dateien möglich wird. Unterhalb welchen Verzeichnisses die Daten welcher Partition zu liegen kommen, kann fest verdrahtet in der Datei `/etc/fstab` festgelegt oder beim manuellen Mounten auf der Kommandozeile angegeben werden.
In der Konfigurationsdatei `/etc/fstab` steht, welche permanenten Dateisysteme existieren und wie sie in den Dateibaum eingehängt werden sollen. Sie wird beim Booten gelesen (und muss daher auf der Root-Partition liegen) und sorgt dafür, dass die Daten auf den übrigen Festplattenpartitionen im Dateibaum zugänglich werden. Die `/etc/fstab` enthält die folgenden Informationen:

- `Device`
  die Gerätedatei des Massenspeichers (z. B. `/dev/fd0` für's Diskettenlaufwerk oder `/dev/hda1` für die erste Partition der ersten IDE-Platte)
- `Mountpoint`
  der Mountpoint, also das Verzeichnis, in dem die Daten nach dem Mounten zu finden sein werden
- `Type`
  der Dateisystemtyp (z. B. `ext2` oder `reiserfs`; oft reicht auch `auto`, das den Kernel bittet, das Dateisystem selbst herauszufinden)
- `Options`
  Mit den – vom zu mountenden Dateisystem abhängigen – Optionen lässt sich der Mount-Vorgang steuern. Eine Auswahl:

  `defaults` Voreinstellungen (rw, suid, auto, nouser ...)

  `noauto` Kein automatisches Mounten beim Booten

[13] http://www.nsa.gov/selinux/





```
for i in *.HTM
```

ein, erscheint ein sogenannter *Second-Level-Prompt*, der von der Umgebungsvariablen `PS2` bestimmt wird:

```
$  echo $PS2
>
```

Dieser erinnert daran, dass das Kommando noch unvollständig ist. Hinter diesem Prompt können wir weiterschreiben

```
> do mv $i `basename $i HTM`html
> done
$
```

und sparen uns so die Semikola. Schreiben wir

```
#!/bin/bash

for i in *.HTM
do
mv $i `basename $i HTM`html
done
```

in eine Datei, der wir Ausführbarkeitsrechte verleihen, haben wir ein kleines Programm, ein `Shellskript`, das sich wiederholt ausführen lässt.
Die erste Zeile des Skripts ist einem Kommentar vorbehalten, der besagt, welcher *Interpreter* den Rest ausführen soll – hier also `/bin/bash`. Natürlich kann frau ein Shellskript auch mit „normalen“ Kommentaren versehen, die der Dokumentation dienen. Einfach ein `#` davor, und schon kümmert sich die Shell nicht mehr um den Rest der Zeile.

✎ Schreib ein kleines Skript, das mit Hilfe von `wc` herausfindet, wieviele Zeilen jede einzelne Datei in Deinem Homeverzeichnis lang ist.

**cut** In Shellskripten lassen sich grundsätzlich alle Kommandozeilenbefehle einsetzen, wobei eine Beschränkung auf Standard-Unixtools wie die in diesem Kurs vorgestellten dann angeraten ist, wenn das Skript auf verschiedenen Installationen laufen soll. (Soll es auch auf anderen Unixsystemen oder gar in einer anderen, von der Bourne-Shell abgeleiteten Shell ausgeführt werden, gilt es zudem, die verwendeten Befehlsoptionen zu überprüfen bzw. sich auf Syntaxelemente der Bash zu beschränken, die auch zum Repertoire der anderen Shell gehören.)
Ein in Shellskripten sehr nützlicher Befehl ist `cut`. Mit ihm kann frau aus dem Text einer Datei oder von der Standardeingabe Spalten extrahieren. Ob dies byte- bzw. zeichengenau oder anhand von Feldern, die von *Delimitern* (Spaltentrennern) begrenzt werden, geschieht, ist abhängig von der angegebenen Option. Ohne explizite Angabe eines Spaltentrenners dienen Tabulatoren als Feldbegrenzer.

```
cut -d ":" -f 1,5 /etc/passwd
```

beispielsweise sucht aus der Passwort-Datei alle auf diesem Rechner vorhandenen Accounts und den dazugehörigen Kommentar heraus.

✎ Schreib ein Shellskript, das für alle Benutzerinnen Deines Rechners eine namentliche Begrüßung ausgibt.

✎ Was macht das Kommando `cut -b 6-17`? Was könnte der Unterschied zu `cut -c 6-17` sein?

Kindprozesse der Shell erben ihr *Environment*, ihre Umgebung. Variablen, die einfach nur so auf der Kommandozeile gesetzt wurden, gehören nicht da hinein und werden daher mit ihren Inhalten auch nicht an Kindprozesse weitergegeben.
Will frau dafür sorgen, dass eine Variable mit dem aktuellen Inhalt auch auf Kindprozesse übergeht, muss sie sie mit dem Befehl `export` *exportieren*. Das haben wir ohne Erklärung schon bei den Umgebungsvariablen gesehen – hier nochmal ausführlich:

```
bash$ variable=wert   # Setzen der Variablen in der aktuellen Shell
bash$ echo $variable
wert
bash$ bash            # Starten einer Kindshell
bash$ echo $variable  # variable wurde nicht an Kindshell exportiert
bash$ exit            # Ausloggen aus der Kindshell
exit
bash$ echo $variable  # zurueck in der Ausgangsshell
wert
bash$ export variable # Exportieren von variable
bash$ bash            # Neue Kindshell
bash$ echo $variable  # Kindshell kennt exportierte variable
wert
bash$ exit
exit
```

**for-Schleife** Es kommt recht häufig vor, dass frau mehrere Dateien gleichartig bearbeiten will. Einfache Kommandos nehmen oft mehrere Dateien als Argumente an, doch wenn die Aufgabe sich nicht mehr mit einem einzigen Kommando lösen lässt, wird es schwierig. Es sei denn, frau besinnt sich auf die `for`-Schleife:

```
for i in *.HTM; do mv $i `basename $i HTM`html; done
```

Die Shell schaut nach, auf welche Dateien im aktuellen Verzeichnis das Wildcard-Muster `*.HTM` passt und legt deren Dateinamen einen nach dem anderen (d. h. bei jedem Schleifendurchlauf einen) in der Laufvariablen `i` ab.
Das war der Schleifenkopf und damit das erste Kommando. Will frau mehrere Kommandos hintereinanderweg auf der Kommandozeile schreiben, muss sie die mit Semikola trennen. Das zweite Kommando wird eingeleitet vom Schleifen-Schlüsselwort `do`, das den Beginn des Schleifenrumpfes markiert. In jedem Schleifendurchlauf soll also der Befehl `mv $i `basename $i HTM`html` ausgeführt werden.
Was soll da umbenannt werden? Offensichtlich die Datei, deren Name gerade in der Variablen `i` steht. Der neue Name endet auf `html`, doch was ist das, was zwischen den *Backticks* ` ` steht? Wenn wir es einzeln anschauen, sieht es gar nicht mehr so fürchterlich aus: `basename $i HTM`. Das Kommando `basename` findet den einfachen Dateinamen seiner Argumentdatei heraus, wobei es sämtliche Pfadangaben wegstreicht. Der Basename von `/etc/passwd` ist zum Beispiel `passwd`. Gibt frau `basename`ein weiteres Argument mit auf den Weg, interpretiert das Kommando dies als Dateinamenendung, die es ebenfalls wegzustreichen gilt. Im Beispiel hackt `basename` also die Endung `HTM` von der Datei in `i` ab, um anschließend eine neue Endung `html` anhängen zu können. Die Backticks sorgen lediglich dafür, dass `basename` zur Sache kommt, bevor das `mv`-Kommando ausgeführt wird.
Zu guter Letzt gilt es, den Schleifenrumpf mit `done` abzuschließen.
Natürlich muss niemand all diese Kommandos auf eine Zeile schreiben. Tippt frau in der Shell





`user` Device darf von normalen Nutzerinnen gemountet werden

`ro, rw` nur lesbar („read only“), les- und schreibbar („read write“)

`exec` Ausführung von ausführbaren Dateien (also Programmen) gestattet

`sync` Ungepuffertes Schreiben

- `Dump`
  Gilt momentan nur für Ext2 und besagt, ob das Dateisystem durch den Befehl `dump` gesichert werden soll. Hat eher selten praktische Bedeutung.

- `Check`
  Gibt an, ob das Dateisystem vor dem Mounten überprüft werden soll. Beim Root-Dateisystem sollte hier eine 1 stehen und bei allen anderen entweder eine 0 (keine Prüfung) oder eine 2. Dateisysteme mit gleicher Nummer werden parallel überprüft, das Root-Dateisystem sollte immer allein und als erstes getestet werden.

Wenn in der `/etc/fstab` Folgendes zum Diskettenlaufwerk steht...

```
/dev/fd0 /floppy vfat noauto,user 0 0
```

..., kann jede Benutzerin eine Diskette durch den Befehl `mount /floppy` mounten bzw. mit `umount /floppy` wieder unmounten. Fehlt diese Zeile, darf nur `root` das Diskettenlaufwerk in den Dateibaum einhängen und muss alle Optionen extra mitgeben.
`mount` kennt u. a. folgende Flags:

`-t` Dateisystemtyp

`-o` Optionen (die gleichen wie in der `/etc/fstab`)

Demnach bindet frau eine Windows-Diskette folgendermaßen in den lokalen Dateibaum unterhalb von `/floppy` ein:

```
mount /dev/fd0 /floppy -t vfat
```

Welche Dateisysteme mit welchen Parametern aktuell gemountet sind, steht in der Datei `/etc/mtab`, die sich bei ihrer Ausgabe an die Syntax der `/etc/fstab` hält. Außerdem liefert der Befehl `mount` Informationen über gemountete Systeme, deren Mountpoints und Mountparameter.

✎ Schau Dir die Datei `/etc/fstab` an und ermittle, welche Arten von Dateisystemen im Rechenzentrum verwendet werden und was damit gemacht werden kann.

Die wichtige Information, wieviel Platz sich aktuell auf einer Partition befindet, gibt übrigens `df` („disk free“).

✎ Wie voll sind die Platten momentan?

## 4.3 Dateien

Linux behandelt nahezu alles als Datei. Dem liegt der Gedanke zugrunde, dass sich gleiche Mechanismen wie z. B. Lesen und Schreiben auf ganz unterschiedliche Dinge anwenden lassen (siehe auch Kapitel 5.1, ab Seite 27).

- *Reguläre Dateien*

- von Menschen (mit entsprechenden Kenntnissen) lesbare Dateien: z. B. Texte, Quellcode von Programmen, HTML-, XML-, PostScript-Dateien ..., die traditionell ASCII-, inzwischen häufig auch UTF-8- (oder zumindest sprachspezifisch wie ISO-8859-15[14]) kodiert vorliegen
- Dateien in anwendungsspezifischen Binärformaten: komprimierte Dateien, Bilddateien, proprietäre Formate ...
- maschinenlesbare (Binär-)Dateien: Programme und Bibliotheken

- *Geräte-Dateien*

  Unter Linux wird auch Hardware über eine Datei als Bindeglied verwaltet. Dabei unterscheiden wir

  - *blockorientierte Gerätedateien*, dazu zählen alle Massenspeicher wie z. B. das Diskettenlaufwerk (`/dev/fd0`) oder eine IDE-Festplatte (`/dev/hda1-x`), und
  - *zeichenorientierte Gerätedateien*, über die z. B. die Maus (`/dev/mouse` bzw. `/dev/cua0-x` bei einer seriellen oder `/dev/psaux` bei einer PS/2-Maus) oder der Drucker (`/dev/lp1`) angesprochen werden.

- *Links*

  Verweise auf schon vorhandene Dateien. Bei den sogenannten *Hard Links* handelt es sich um einen weiteren Dateinamen für die ursprüngliche Datei. Ein File verschwindet erst physikalisch, wenn sowohl die ursprüngliche Datei als auch alle auf sie verweisenden *Hard Links* gelöscht sind. Anders die *Soft Links* oder *symbolischen Links*: Sie verweisen lediglich auf den ursprünglichen Dateieintrag und gehen ins Leere, wenn die Datei, auf die sie zeigen, entfernt wurde. Während symbolische Links auch auf Verzeichnisse und über Partitionsgrenzen hinaus verweisen dürfen, bleiben Hard Links auf eine Partition und Nicht-Verzeichnisse beschränkt.

- *Verzeichnisse*

  Auch Verzeichnisse werden wie Dateien behandelt. Das aktuelle Verzeichnis versteckt sich z. B. hinter der Datei namens `.` (Punkt), das darüberliegende Verzeichnis heißt `..` (Punktpunkt).

- *FIFO*s

  („First In First Out“) oder *Named Pipes* sind Dateien, die es ermöglichen, dass zwei Prozesse untereinander durch Pipes (ähnlich denen im Abschnitt 11.1 auf Seite 54 beschriebenen) miteinander kommunizieren können.

- *Sockets*

  Bieten eine ausgefeiltere Möglichkeit, über die lokale Client- und Server-Prozesse untereinander kommunizieren können. Beispiele dafür finden KDE-Nutzerinnen in den Verzeichnissen `/tmp/mcop-`*username* und `/tmp/ksocket-`*username*.

[14] umfasst grob gesagt ASCII plus Umlaute u. a. akzentuierte Buchstaben plus Eurozeichen





| | |
|---|---|
| `cat p*[1-3]` | zeigt den Inhalt aller Dateien an, die mit einem kleinen `p` beginnen und mit einer `1`, `2` oder `3` enden. |

✎ Suche eine Kombination, die Deinen und den Namen Deiner rechten Nachbarin matcht, aber den Deiner linken Nachbarin nicht.

## 11.2 Shellskripte

Wildcards, Pipes, Ein- und Ausgabeumlenkung sind aber noch nicht alles, was das Powertool Shell zu bieten hat – als „Killerfeature“ haben Unixshells alles eingebaut, was frau von einer Programmiersprache erwartet: Variablen, konditionale Abfragen, Schleifen.

**Variablen** Eine Variable in der Bash zu setzen ist ganz einfach: Frau denkt sich einen Namen für sie aus und setzt hinter ein Gleichheitszeichen den Wert. Um ihren Inhalt herauszufinden, gilt es, ein Dollarzeichen vor den Variablennamen zu setzen. Will frau den Inhalt ausgeben, benutzt sie – wie wir schon beim Spezialfall Umgebungsvariable gesehen haben – das Kommando `echo`:

```
bash$ variable=wert
bash$ echo $variable
wert
```

Da Leerzeichen von der Shell als Worttrennzeichen benutzt werden, müssen sie geschützt werden, wenn sie Bestandteil einer Zeichenkette sind, die einer Variable zugewiesen wird. Dabei gibt es die „sanfte“ Möglichkeit, den gesamten String in doppelte Anführungszeichen (*Doppelquotes*) zu setzen, oder die unnachgiebige Variante mit einfachen Anführungszeichen. Bei ersterer dient der Dollar weiterhin zum Herausfischen von Variableninhalten, in letzteren interpretiert ihn die Shell buchstäblich:

```
bash$ variable="wert2 $variable"
bash$ echo $variable
wert2 wert
bash$ variable='wert2 $variable'
bash$ echo $variable
wert2 $variable
```

✎ Warum reagiert die Shell jetzt auf den Befehl `variable=wert2 $variable` mit `bash: wert2: command not found`?

✎ Warum schweigt die Shell auf die Befehlsfolge `unset variable; variable=wert2 $variable` hin, statt die oben genannte Fehlermeldung auszugeben?

Das Verhalten der Bash und anderer Programme wird nicht nur durch Umgebungs-, sondern auch durch *Shellvariablen* beeinflusst. Letztere werden von der Shell selbst genutzt und können mit dem Kommando `set` angezeigt werden. Darunter fallen zum Beispiel die Variable `IFS`, in der festgelegt ist, welche Zeichen als Worttrennzeichen fungieren, oder `PS1`, die die Eingabeaufforderung (den *Prompt*) festlegt. Die Umgebungsvariablen erfährt frau mit dem Kommando `env`.

✎ Wie ist der Prompt bei Dir definiert? Sieh in der Manpage zu `bash` nach, welche Platzhalter sich darin verwenden lassen. Ändere Deinen Prompt temporär in `aktuelles_verzeichnis(rechnername)$!`

`editor` und `EDITOR` zwei völlig verschiedene Dinge sind: Für `editor` interessieren sich im Gegensatz zur vordefinierten Umgebungsvariablen `EDITOR` höchstens von der Benutzerin geschriebene Skripte.
Zum Ausgeben eines einzigen Variableninhalts dient das Kommando

```
echo $variable
```

Dabei muss ein `$`-Zeichen direkt vor dem Variablennamen stehen.

```
echo $PATH
```

gibt den Pfad aus, in dem nach ausführbaren Dateien gesucht wird. Wenn frau jetzt ein zusätzliches Verzeichnis (beispielsweise `bin` im Homeverzeichnis) in den Pfad mit einbinden will, so kann sie das durch Neudefinition der Umgebungsvariablen tun, zum Beispiel durch das Kommando

```
export PATH=$HOME/bin:$PATH
```

Das abschließende `:$PATH` stellt sicher, dass der alte Pfad zusätzlich erhalten bleibt; die Umgebungsvariable `HOME` enthält den Pfad des eigenen Homeverzeichnisses. Wenn frau diese Zeile eingibt, gilt der neue Pfad dank `export` zwar für alle Programme, die aus dieser Shell heraus aufgerufen werden, aber abgesehen davon natürlich nur für die aktuelle Shell in der aktuellen Sitzung. Er verbreitet sich nicht automatisch unter allen anderen, bereits laufenden Shells, und beim nächsten Einloggen ist er vergessen. Soll letzteres nicht geschehen, trägt frau das Kommando als zusätzliche Zeile in die Datei `.bashrc` und/oder `.bash_profile` im eigenen Homeverzeichnis ein. Der so gesetzte Suchpfad wird von der Shell übrigens in genau der Reihenfolge durchsucht, in der die Verzeichnisse darin stehen. Ein Programm namens `ls` in `$HOME/bin` würde also bedeuten, dass bei einem `ls`-Aufruf niemals das System-Programm `/bin/ls` zum Zuge käme. Dieses müsste frau dann – genau wie alle nicht im Suchpfad stehende Programme – mit Pfadangabe als `/bin/ls` aufrufen.

✎ Sieh Dir Deinen Suchpfad an und erweitere ihn um ein Verzeichnis mit ausführbaren Dateien, das bislang nicht enthalten ist (z. B. `/usr/sbin`). Starte eins der darin enthaltenen Programme (z. B. `/usr/sbin/lsof`) mit und ohne Pfad. Versuche dasselbe in einer anderen Shell.

**Wildcards** Soll sich ein Kommando auf mehrere Dateien beziehen, so lässt sich eine Menge Schreibarbeit sparen, indem sogenannte *Wildcards* („Platzhalter“) zum Einsatz kommen. Die wichtigsten sind

| | |
|---|---|
| `*` | steht für eine beliebige Zeichenkette, |
| `?` | steht für ein einzelnes Zeichen, und |
| [*Zeichenmenge*] | steht für eines der in Klammern angegebenen Zeichen. |

Ein paar Beispiele:

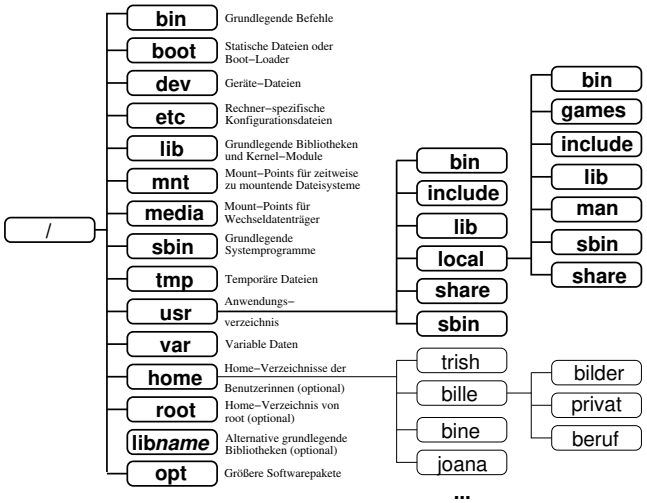
| | |
|---|---|
| `cat p*` | zeigt den Inhalt aller Dateien an, die mit einem kleinen `p` beginnen, inklusive einer eventuellen Datei `p`. |
| `cat p?` | zeigt den Inhalt aller zweibuchstabigen Dateien an, die mit einem kleinen `p` beginnen. |
| `cat [pP]*` | zeigt den Inhalt aller Dateien an, die mit einem kleinen `p` oder einem großen `P` beginnen |





## 4.4 Verzeichnisbaum

Alle Dateien zusammen ergeben den Verzeichnisbaum, dessen oberstes[15] Verzeichnis ein „/“ symbolisiert. Hier kommen die Dateien aus der Root-Partition zu liegen; der Ordner selbst wird als *Wurzel-Verzeichnis* oder *Root Directory* bezeichnet. In darin befindlichen Ordnern („Mountpoints“) werden ggf. andere Partitionen eingehängt. Das erklärt, warum sich auf einer Partition immer nur ein Teilast des Dateibaums befinden kann.
Die Dateien im Linux-Dateibaum sind nach ihren Aufgaben sortiert; aus Sicht der Benutzerin ist nicht erkennbar, auf wieviele Partitionen sie sich verteilen.
Leider hat die Existenz von Distributionen die Folge, dass sich Dateien gleicher Funktion auf verschiedenen Systemen an unterschiedlichen Stellen befinden. Um diesen Missstand zumindest teilweise zu beheben, haben sich die großen Distributoren und andere Software-Hersteller zusammengeschlossen und entwickeln die LSB („Linux Standard Base“), einen Standard, der grundlegende Festlegungen zum Aufbau eines Linux-Systems trifft. Dort ist unter anderem im FHS („Filesystem Hierarchy Standard“) festgelegt, wie der Verzeichnisbaum einer Linux-Installation aussehen sollte:




Die unterhalb des Wurzel-Verzeichnisses sichtbare Aufteilung der Systemsoftware in Programme (`/bin` und `/sbin`), Konfigurationsdateien (`/etc`) und Bibliotheken (`/lib`) wiederholt sich unterhalb des `/usr`-Zweigs. Hier kommen die Anwendungsprogramme und andere, für das System nicht lebenswichtige Software-Pakete unter. Da diese Dateien mitbringen, die nicht in besagte Kategorien passen, gibt es hier neue Verzeichnisse wie `share` für architekturunabhängige Dateien wie Icons, Wörterbücher, Manpages (`/usr/share/man`) etc. oder `include` für sogenannte Headerdateien, die Schnittstellenbeschreibungen für Bibliotheken enthalten und benötigt werden, wenn frau selbst Software kompilieren will.

[15] ... bzw. unterstes, je nachdem, ob frau die Wurzel als „unten“ oder „oben“ bezeichnet; bei Dateibäumen hat es sich aber durchgesetzt, die Wurzel bei Darstellungen oben hin zu legen.

Programme und andere Dateien, die die Distribution nicht selbst mitliefert, gehören ins Verzeichnis `/usr/local`, das die Unterstruktur von `/usr` im Wesentlichen wiederholt. Damit lassen sich Probleme beim Update/Upgrade vermeiden.
Der FHS legt fest, dass im `/usr`-Zweig des Dateisystems (außer zum Zwecke der Software-Installation) nichts geschrieben werden darf. Will ein Programm sich während der Laufzeit Daten merken, gehören die nach `/tmp`, `/var` oder in die Home-Verzeichnisse der Nutzerinnen. Auch für `var` schreibt der FHS eine bestimmte Unterstruktur vor, die jedoch erst interessant wird, wenn frau in die Tiefen der Systemadministration einsteigen will. Gut zu wissen ist hier, dass das Verzeichnis `/var/log` Protokolldateien („Log-Dateien") von Serverdiensten beherbergt, die bei der Fehlersuche behilflich sein können.

## 4.5 Bewegen im Verzeichnisbaum

**Verzeichnis wechseln** Um ein Verzeichnis zu wechseln, verwendet frau den Befehl *cd* („change directory"). Der Befehl alleine katapultiert sie in ihr Homeverzeichnis. Normalerweise wird er aber benutzt, um in einen bestimmten Ordner zu wechseln. Die Adresse dieses Verzeichnisses, also sein Pfad, „der Weg dahin", kann auf die beiden folgenden Weisen angegeben werden:

**Relativer Pfad:** Die Angabe des Zielverzeichnisses erfolgt vom momentanen Standpunkt aus. Das Oberverzeichnis dazu symbolisieren zwei Punkte (`cd ..`), Unterverzeichnisse spricht frau einfach mit ihren Namen an (`cd mail`). Die Angaben dürfen durchaus auch verschachtelt werden, z. B. zu `cd ../../usr/src`.

**Absoluter Pfad:** Dabei geht die Benutzerin vom "höchsten Punkt" im Dateibaum, dem Root- oder Wurzel-Verzeichnis, aus: `cd /usr/src`. Den absoluten Pfad erkennt sie an einem führenden Slash (/).

**Auflisten des Verzeichnisinhalts** Zum Ansehen des Inhalts eines Verzeichnisses dient der Befehl `ls` („list"). Er listet ohne Argument alle im aktuellen Verzeichnis vorhandenen Dateien auf. Bekommt `ls` ein Argument mit auf den Weg, so zeigt das Kommando den Inhalt dieses Verzeichnisses an.
Außerdem können noch Optionen mit dazugepackt werden, die die Art der Augabe steuern. Eine kleine Auswahl:

- Auch versteckte (mit einem Punkt beginnende, vgl. Seite 55) Dateien auflisten („all"): `ls -a`
- Dateirechte, -größe und Eigentumsverhältnisse aufführen („long"; vgl. Seite 27): `ls -l`
- Dateiattribute anzeigen: `ls -F`

  Bei letzterer Option kommen folgende Markierungen zum Einsatz:

  / Verzeichnis,
  * ausführbare Datei,
  @ Link

✎ Handelt es sich bei der Datei `xmag` im Verzeichnis `/usr/X11R6/bin/` um ein Verzeichnis, einen Link, ein Programm oder eine normale Datei?





Programme, die ihre Eingabe von stdin bekommen, Operationen mit diesen Eingaben ausführen und das Ergebnis nach stdout schreiben, heißen *Filter*. Beispiele sind `grep`, `cat`, `more`, `sort` und `wc` („word count").

✎ Wieviele (rpm-)Pakete sind auf Deinem Rechner installiert? Nutze dazu eine Pipe und das Kommando `wc -l`.

✎ Wie ließe sich diese Aufgabe ohne Pipe, aber dafür mit einer Aus- und einer Eingabeumlenkung in eine Datei lösen?

✎ Worin unterscheidet sich die Ausgabe von `wc -l datei` von der Variante mit Eingabeumlenkung und warum?

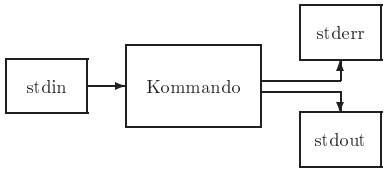
✎ Suche mit einer einzigen Kommandozeile die Namen aller Dateien und Verzeichnisse in Deinem Homeverzeichnis heraus, die heute verändert wurden, und speichere das Ergebnis in einer neuen Datei namens `heute` ab. Benutze dazu die Kommandos `ls` (mit passenden Optionen), `grep` und `cut -c 57-`, Pipes und eine Ausgabeumlenkung.

✎ Was tut `cut -c 57-`?

✎ Überprüfe, ob die Anzahl der Verzeichniseinträge im Verzeichnis `/usr` tatsächlich mit der Zahl im zweiten „`ls -al`"-Block übereinstimmt! (Vgl. Seite 27ff.) Zum Zusammenzählen der Zeilen („lines") kann das Kommando `wc -l` dienen.

**Startdateien** Ähnlich wie unter DOS, wo beim Starten des Rechners die Kommandos aus den Dateien `autoexec.bat` und `config.sys` ausgeführt, werden auch beim Einloggen in ein Unix-System bestimmte Dateien abgearbeitet und Standardeinstellungen gesetzt.
Da ist zunächst die Datei `/etc/profile`, die von der Systemadministratorin gewartet wird und auf die normale Benutzerinnen keinen Schreibzugriff haben. Beim Einloggen auf der Textkonsole mit der Bash als Login-Shell (oder wenn eine Bash explizit als Login-Shell gestartet wird) werden anschließend die Dateien `.bash_profile`, `.bash_login` und `.profile` im Homeverzeichnis der Benutzerin abgearbeitet. Somit sind die Einstellungen festgelegt, die für die gesamte Sitzung gelten sollen. (Es sei denn, die Benutzerin ändert sie während der Sitzung ausdrücklich.)
Bei interaktiven Nicht-Login-Shells lädt die Bash die Session-bezogenen Voreinstellungen aus der Datei `.bashrc`, so im Homeverzeichnis der Benutzerin vorhanden. Diese Datei wird jedesmal ausgewertet, wenn eine neue Bash aufgerufen wird – sei es durch das Kommando `bash` oder das Öffnen eines neuen X-Terminal-Fensters[35].
Dateien wie `.profile` oder `.bashrc`, die mit einem Punkt beginnen, heißen auch *versteckte Dateien*, da sie normalerweise vom Kommando `ls` nicht angezeigt werden. Es handelt sich dabei in der Regel um Konfigurationsdateien für verschiedene Unix-Programme, die im Normalbetrieb meistens weniger von Interesse sind und die anzuzeigende Liste nur unnötig vergrößern.
Die o. g. Systemdateien setzen u. a. *Umgebungsvariablen*, die – wie zum Beispiel `EDITOR` – bestimmte Voreinstellungen für die Arbeitsumgebung treffen. Frau kann sie sich mit dem Befehl `env` anzeigen lassen. Die Namen der *Environment-Variablen* werden laut Konvention groß geschrieben; prinzipiell spielt die Schreibung von Variablennamen aber nur insofern eine Rolle, als dass

[35] Da frau jedoch jede Shell mit dem Flag `-login` zu einer Login-Shell machen kann, können auch über Icons/Menüs gestartete Shells Login-Shells sein. Dies hängt in der werksseitigen Voreinstellung von der Vorliebe des Distributors ab und kann die Benutzerin schon mal zur Verzweiflung treiben, wenn die von ihr gewählte Start-Datei partout nicht ausgeführt wird.




- *stdin*, die Standardeingabe,
- *stdout*, die Standardausgabe, sowie
- *stderr*, die Standardfehlerausgabe.

Die Standardeingabe ist üblicherweise mit der Tastatur verbunden, damit die Benutzerin überhaupt etwas eingeben kann. Standardausgabe und Standardfehlerausgabe „hängen“ am Bildschirm, weshalb Datenoutput und Fehlermeldungen auf dem Monitor erscheinen. Dabei spricht die Shell Tastatur und Bildschirm – wie schon mehrfach erwähnt – über Dateien an. Da es einem Programm normalerweise vollkommen egal ist, aus welcher Datei die Eingaben gelesen werden bzw. in welche Datei die Fehlermeldungen oder die Ausgabe geschrieben werden, lassen sich die Kommunikationskanäle problemlos auf andere Dateien umbiegen:

| | |
|---|---|
| *Unixkommando* `>` *Ausgabedatei* | lenkt stdout in Ausgabedatei um. |
| *Unixkommando* `>>` *Ausgabedatei* | lenkt stdout in Ausgabedatei um und hängt die Information an eine eventuell bestehende Datei an. |
| *Unixkommando* `<` *Eingabedatei* | sorgt dafür, dass stdin aus der *Eingabedatei* liest. |
| *Unixkommando* `2>` *Fehlerdatei* | lenkt die Fehlermeldungen in eine Datei um. |

In der Praxis sieht das so aus:

| | |
|---|---|
| `ls -lF > xxx` | Die erzeugte Liste kann anschließend in `xxx` besichtigt werden. |
| `mkdir test; rm test 2>fehler` | In der Datei `fehler` steht jetzt die Fehlermeldung, dass `test` ein Verzeichnis ist und folglich nicht mit `rm` gelöscht werden kann[34]. |

Zur Abrundung dieses Themas zwei häufig benutzte Umlenkungsbeispiele:

- `2>/dev/null` leitet die Fehlerausgabe des Programms ins Nulldevice, also ins Nichts.
- `2> &1` leitet die Fehlerausgabe auf die Standardausgabe um.

**Die Pipe** Häufig möchte frau die Ausgabe eines Kommandos als Eingabe für das nächste Kommando verwenden, ohne dass die Zwischenergebnisse in Form einer Datei hinterher noch benötigt werden. Dafür bietet Unix die *Pipes* (Röhren, Rohrleitungen). Damit leitet frau die Standardausgabe eines Kommandos direkt auf die Standardeingabe des nächsten Kommandos um. Als Zeichen für die Pipe dient der senkrechte Strich: `|`

```
ls -lF /dev | more
grep bash /etc/passwd | less
```

Es ist möglich, beliebig viele Kommandos durch Pipes zu verbinden („pipeline“). Voraussetzungen zum Aufbau einer Pipe sind: Das erste Programm schreibt seine Ausgabe nach stdout, das zweite Programm liest seine Eingaben von stdin.

[34] Das Semikolon in dieser Befehlszeile wirkt wie ⏎. So kann frau mehrere, hintereinander auszuführende Kommandos in einer Kommandozeile unterbringen.





## 5 Benutzerinnen und ihre Rechte

Linux ist ein Multiuserinnensystem, darauf optimiert, in Mehrbenutzerinnen-Umgebungen zu laufen. Daher existieren auf einem Linuxrechner immer unterschiedliche „Konten“, *Accounts*, für Benutzerinnen, die in Bezug auf die im Dateibaum enthaltenen Dateien unterschiedliche Rechte haben.
Das hat vor allem sicherheitstechnische Gründe. Da die Dateien Besitzerinnen „gehören“, führt ein Bedienungsfehler einer unprivilegierten Benutzerin nicht dazu, dass das ganze System nicht mehr funktioniert: Sie hat nämlich (im Allgemeinen) gar keinen Zugriff auf Systemdateien.
Die einzige Benutzerin im System, die die Macht zu solcher Zerstörung hat, heißt `root`. Sie darf alles[16] – und sollte aus diesem Grund wirklich nur Verwaltungaufgaben nachgehen. Wer grundsätzliche mit `root`-Rechten arbeitet, macht sein System ähnlich anfällig für Trojaner, Viren und Co., wie es Windows-Systeme sind – mit dem kleinen Unterschied, dass es derzeit praktisch keine Viren für Linux gibt.
Stattdessen nutzt frau zum Arbeiten (und Surfen) das Konto einer nichtprivilegierten Benutzerin. Neben diesen Accounts, die Personen zugeordnet werden, existieren eine Reihe virtueller Pseudo-User, deren Konten nur dafür da sind, dass Server u. a. im Hintergrund laufenden Programme mit diesen statt mit `root`-Rechten ausgestattet werden. Damit bietet das System weniger Angriffsfläche nach außen.
Die LSB sieht als Mindestbelegschaft eines Linux-Systems die User `root` (Systemverwalterin mit allen Rechten), `bin` (Systemverwalterin mit eingeschränkten Rechten) und `daemon` (Account zur Unterprozessverwaltung) vor. Darüber hinaus schlägt der Standard eine Reihe weiterer Systembenutzerinnen für unterschiedliche Aufgaben vor. Viele dieser „Pseudo-User“ haben nicht einmal ein eigenes Homeverzeichnis.

### 5.1 Dateirechte

Bei Linux werden für alle Dateien Rechte vergeben, die regeln, wer was mit den Dateien anstellen darf. Der Befehl, um sich diese Informationen zu Gemüte zu führen, lautet `ls`. Mit den Optionen `-al` versehen zeigt das Kommando die üblichen Dateirechte aller Dateien im Verzeichnis an. Um `ls` zu überreden, nur die Eigenschaften des als Argument angegebenen Verzeichnisses (und nicht dessen Inhalt) aufzulisten, gibt es die Option `-d`[17].
Die folgenden Zeilen zeigen, wie die Ausgaben von `ls -al` für unterschiedliche Dateitypen aussehen:

```
-rw-r--r--   1 bille  users  236472 Sep  2 21:42 /home/bille/inf/lin.tex
lrwxrwxrwx   1 bille  users      10 Sep  2 22:50 www -> /home/www/
prw-r--r--   1 bille  users       0 Sep  2 22:53 named_pipe
brw-rw----   1 root   disk   2,   0 Jan 19  2000 /dev/fd0
brw-rw----   1 root   disk   2,   1 Jan 19  2000 /dev/fd1
crw-rw----   1 root   lp     6,   0 Jan 19  2000 /dev/lp0
drwxr-xr-x  29 root   root     4096 Mär 15 21:47 home
drwxr-xr-x  21 root   root     4096 Mai 28 09:24 .
drwxr-xr-x  21 root   root     4096 Mai 28 09:24 ..
```

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨

✎ Worauf beziehen sich die letzten zwei Beispielzeilen zu `.` und `..`?

[16] Es gibt Erweiterungen wie RSBAC, die es erlauben, Systeme aufzubauen, in denen auch `root` nicht alles darf, doch die sind derzeit nicht Bestandteil des offiziellen Linux-Kernels.
[17] Weitere Optionen siehe `man ls`.

Aufgeteilt in neun Blöcke präsentiert `ls` folgende Informationen:

① Dateityp und erlaubte Aktionen. Diesen Block betrachten wir im folgenden Abschnitt näher.

② Anzahl der Dateinamen (ursprünglicher Name und Hard Links) bei Dateien, Anzahl der Unterverzeichnisse bei Verzeichnissen.

③ Besitzerin.

④ Gruppe.

⑤ Dateigröße bei regulären Dateien bzw. Minor- und Major-Number bei Gerätedateien. Die Major-Number definiert die Art des Geräts, (z. B. 2 für das Diskettenlaufwerk (im Beispiel `/dev/fd0` und `/dev/fd1`) und 6 für den Drucker (`/dev/lp0`); bei Anforderungen an dieses Device sagt sie dem Kernel im allgemeinen, welchen Treiber er benutzen soll. Die Minor Number wiederum legt fest, um das wievielte Gerät dieser Sorte es sich handelt.

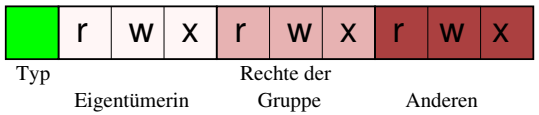
⑥ Monat der letzten Änderung.

⑦ Tag der letzten Änderung.

⑧ Uhrzeit bzw. Jahr (wenn die Datei älter als ein Jahr ist) der letzten Änderung.

⑨ Dateiname.

Block Nummer 1 besteht aus zehn Spalten zu je einem Zeichen. Die erste davon bezeichnet den Dateityp, die nächsten drei Spalten zeigen die Rechte der Dateibesitzerin, die drei weiteren die Rechte der Gruppe und die letzten drei die Rechte aller anderen Nutzer des Systems:




Folgende Dateitypen kommen infrage:

`-` Reguläre Datei

`d` Verzeichnis

`l` Symbolischer Link

`c` Zeichenorientiertes Gerät

`b` Blockorientiertes Gerät

`s` Socket

`p` Named Pipe

Die zu vergebenden Rechte sind für alle drei Personenkategorien gleich:

`r` Lesen

`w` Schreiben (Verändern), Löschen

`x` Ausführen (bei Programmen), Betreten (bei Verzeichnissen)



## Sonntag, 04. September 2005

## 11 Die Bash

Als wir uns in Kapitel 2.2 mit der Shell beschäftigten, war eine der Aussagen die, dass Unix-Shells allgemein, speziell aber die Bash, die unter Linux die größte Rolle spielt, sehr mächtig seien. Einfaches Kommandos-Eintippen und Prozesse-in-den-Vorder-oder- Hintergrund-Schieben kann es da nicht gewesen sein.

Dieses Kapitel stellt einige Konzepte vor, die der Benutzerin viele Möglichkeiten an die Hand geben, allerdings auch Übung erfordern.

### 11.1 Eingebaute Befehle und Hilfsmittel

**Tipphilfen** Wie hieß der Befehl doch gleich? Irgendwas mit l... Ob frau nun zu faul zum Tippen oder einfach vergesslich ist – die auf Seite 17 beschriebene *Tab-Komplettierung* für Befehle im Suchpfad der Shell ist eines der Features, mit denen die Bash auch gegenüber älteren Unix-Shells punktet.

Die gute Nachricht: Tab-Komplettierung funktioniert nicht nur bei Kommandos, sondern auch bei Verzeichnissen und Dateiangaben.

✎ Versuche, bei der Eingabe des Befehls `mkdir /tmp/bla` mit so wenigen Buchstaben wie möglich auszukommen!

Als weitere Hilfe für die geplagte Anwenderin verfügt die Bash über einen sogenannten *History-Mechanismus*. Damit „merkt“ sie sich ihre „Geschichte“, sprich: alle abgeschickten Kommandozeilen. Mit Hilfe der *Pfeiltasten* ↓ und ↑ kann frau diese Kommandos „wiederverwenden“.

Kommandos aus der History lassen sich auf beliebig komplexe Weise heraussuchen und zweitverwerten. Ein nützliches Beispiel ist die Tastenkombination Esc ., die das letzte Argument des letzten Kommandos auf die Kommandozeile holt. Sie erspart beispielsweise dann eine Menge Tipparbeit, wenn frau mehrere Aktionen mit einer Datei durchführen will.

✎ Nehmen wir an, der letzte Befehl sei `chmod a+x einladung aufraeumen` gewesen. Was passiert, wenn wir nun . / Esc . ↵ eingeben?

**Ein- und Ausgabekanäle** Die Flexibilität der Unix-Kommandozeile hängt vor allem aber auch damit zusammen, dass sich Unix-Kommandos kombinieren lassen. Damit das funktioniert, verfügen sie normalerweise über drei Kommunikationskanäle:

`reboot` wirkt wie ein (sauberer) Warmstart, und tatsächlich ist auf vielen Linuxrechnern der „Affengriff“ (Strg)+(Alt)+(Entf) mit diesem Befehl belegt und auch nichtprivilegierten Usern zugänglich. Ob dem so ist, legt die `/etc/inittab` fest:

```
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

Sicher ausschalten kann frau den Rechner dann, wenn er beginnt, wieder hochzufahren.
Wer seinen Rechner nicht alleine nutzt, sollte sich des Befehls `shutdown` zum Herunterfahren oder Rebooten bedienen. Er ermöglicht es, eine Zeitspanne bis zum tatsächlichen Herunterfahren anzugeben, sodass den Mitbenutzerinnen Zeit bleibt, ihre Arbeit schnell zu beenden. Sie werden automatisch durch eine *Broadcastmessage* davon informiert, dass der Rechner in x Sekunden vom Netz geht und herunterfährt.
So rebootet

```
shutdown -r now "Maschine rebootet jetzt"
```

den Rechner augenblicklich. Alle eingeloggten User erhalten die Nachricht *Maschine rebootet jetzt* angezeigt. `shutdown -h -t 600` fährt den Rechner in 10 Minuten herunter.

✎ Wie würdest Du lediglich eine Warnung an alle eingeloggten User schicken, ohne den Countdown tatsächlich einzuleiten? Nimm die Manpage zu `shutdown` zu Hilfe!





Anstelle des `x`-Bits kann bei Besitzerin und Gruppe das `s`-Bit („set UserID – sUID“) stehen. Steht es in der vierten Spalte, so wird das entsprechende Programme unabhängig von der Aufrufenden mit den Rechten seiner Besitzerin gestartet. Auf diese Weise können beispielsweise auch normale Benutzerinnen mit `passwd` ihr Passwort ändern, obwohl nur `root` die Passwortdatei verändern darf. Andererseits stellen gerade *sUID-root*-Programme eine Gefahr für die Sicherheit des Systems dar: Enthalten diese unsauber programmierte Stellen, über die eine Angreiferin die nur begrenzt verliehenden `root`-Rechte ausweiten kann, ist das mehr als kritisch.
Steht das `s`-Bit anstelle des Gruppen-`x`-Bits, hat das Auswirkung auf Verzeichnisse: Egal wer eine Datei in ein solches Verzeichnis schreibt, sie gehört der Gruppe, die auch das Verzeichnis besitzt. Man nennt dieses Bit das sGID-Bit („set GroupID“).
Im Rechte-Tripel der „anderen“ gibt es zwar kein `s`-, aber dafür ein `t`-Bit („Sticky Bit“, klebriges Bit). Steht es bei Verzeichnissen anstelle des `x`-Bits, so dürfen dort befindliche Dateien nur von deren jeweiliger Besitzerin gelöscht werden, auch wenn alle anderen Nutzerinnen darin ebenfalls Schreibrechte haben. Es kommt zum Beispiel bei Verzeichnissen für temporäre Daten, speziell `/tmp`, zum Einsatz.
Ein großes `S` bzw. `T` findet frau übrigens immer dort, wo kein `x`-Bit für die entsprechende Nutzerinnenkategorie vergeben wurde.

✎ Schau Dir Dein Homeverzeichnis mit `ls -al` an und erkläre, welche Dateien und Verzeichnisse Du darin mit welchen Rechten findest. Welche davon darf Deine Nachbarin lesen? Darf sie das aufgrund der Eigentümerinnen-, Gruppen- oder der Rechte der anderen?

## 5.2 Dateirechte ändern

**Zugriffsrechte** Wenn wir morgen ein kleines Shellskript schreiben, ist das zunächst eine „ganz normale Datei“, die die systemüblichen Dateirechte bekommt, wenn wir sie das erste Mal speichern.
Da wir das Skript wie ein Programm mit seinem Namen (und ggf. dem Pfad) aufrufen wollen, muss es jedoch ausführbar sein. Dazu ändern wir die Dateirechte, indem wir genau angeben, welcher Benutzerinnenkategorie wir welche Rechte zuteilen.
Dazu können wir auf zwei unterschiedliche Weisen vorgehen:

- Durch Angabe
  - der Benutzerinnenkategorie (`u` für die Besitzerin („user“), `g` für die Gruppe, `o` für die Anderen („others“) oder `a` für alle)
  - der Änderungs- (`+` oder `-`) bzw. Zuweisungsoperation (`=`)
  - des Zugriffsrechts (`r`, `w`, `x`, `s` oder `t`)

  ```
  chmod ug+x kleinesSkript
  ```

  ändert die Rechte unseres Skripts für uns und unsere Gruppe auf ausführbar.

  ```
  chmod o-r kleinesSkript
  ```

  entzieht allen anderen die Leserechte.
- Durch Angabe einer dreistelligen Oktalzahl, die sich folgendermaßen ergibt:
  - `1` für das `x`-Bit (Ausführbarkeit),

- 2 für das `w`-Bit (Schreibbarkeit),
- 4 für das `r`-Bit (Lesbarkeit)

Die Rechte berechnet frau für jede Nutzerinnenkategorie einzeln und schreibt sie hintereinander:

```
r w x  r - x  - - -
4+2+1  4+0+1  0+0+0
= 7    = 5    = 0
```

Soll die Datei `kleinesSkript` die angegebenen Rechte bekommen, lautet das passende Kommando

```
chmod 750 kleinesSkript
```

Natürlich lassen sich auch die `s`- und das `t`-Bit auf diese Weise setzen. Dazu wird dem oktalen Tripel eine weitere Stelle vorangestellt, die sich durch Addition der Werte

- 1 für das `t`-Bit (sticky Bit),
- 2 für das sGID-Bit,
- 4 für das sUID-Bit

ergibt.

✎ Das Verzeichnis `informatica` besitzt aktuell die Zugriffsrechte `rwxr-xr-`. Wie ändern sich diese mit dem Kommando `chmod u=rwx,g+w,o=xt informatica`? Welche Auswirkungen hätte ein anschließendes `chmod a=rwx,o+t informatica`?

✎ Erzeuge mit dem Befehl `touch rechte` eine Datei `rechte` in Deinem Homeverzeichnis. Welche Zugriffsrechte hat sie? Benutze die oktale Version, um niemanden außer Dich auf die Datei zugreifen zu lassen. Du selbst verleihst Dir Lese- und Schreibrechte.

✎ Welches Ergebnis hat der Befehl `chmod 5771 datei`?

✎ Probiere aus, was passiert, wenn Du Dir von einer Dir gehörenden Datei und einem Verzeichnis alle Rechte entziehst!

Die Grundeinstellung, mit der zunächst alle neuen Dateien gespeichert werden, legt das Kommando `umask` fest. Als Argument bekommt es die sogenannte „User-Maske“ mit auf den Weg. Subtrahiert frau diese von einer Maximalzahl, erhält sie die Oktalrepräsentation der erwünschten Rechte. Dieser Maximalwert beträgt bei Verzeichnissen 777, bei Dateien 666. Eine umask von 022 (bei den meisten Systemen in der `/etc/profile` voreingestellt) ergibt somit die Rechte 755 für Verzeichnisse und 644 für Dateien.

✎ Ändere Deine umask so, dass Deine neuen Dateien niemand mehr außer Dir lesen kann und dass in neue Verzeichnisse nur Du selbst wechseln darfst.





administratorin spezielle Dinge konfiguriert, die beim Booten der Maschine eingestellt werden sollen.
Wer ganz genau wissen will, wie sich die eigene Distribution diesbezüglich verhält, muss sich unter Umständen Zeile für Zeile durch die `/etc/inittab` arbeiten und durch die jeweils aufgeführten Boot-Helfer- und Init-Skripte wühlen.
Am Schluss des Bootprozederes (die entsprechenden Meldungen können beim Booten in der Konsole normalerweise beobachtet werden), steht das ebenfalls in der `inittab` spezifizierte Start der *virtuellen Konsolen* oder TTYs (daher der Name des entsprechenden Programsm, „get ttys“):

```
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
[...]
```

Im Beispiel wird dazu in den Runleveln 2–5 der Befehl `/sbin/getty 38400 tty1` ausgeführt, in den Runleveln 2 und 3 zusätzlich der Befehl `/sbin/getty 38400 tty2`[33].
Diese starten das `login`-Programm, auf dass frau einen *Login-Prompt* bekommt, an dem sie Username und Passwort eingibt. Auf einer der (mit der Tastenkombination Strg+Fx bzw. von der grafischen Oberfläche Alt+Strg+Fx zu erreichenden virtuellen Konsolen wird in einem Runlevel oft der *Display-Manager* einer grafischen Oberfläche gestartet (`xdm` bzw. sein KDE-Pendant `kdm` oder das entsprechende GNOME-Programm `gdm`), sodass frau sich gleich grafisch einloggen kann. In Runleveln, in denen das nicht vorgesehen ist, startet sie die grafische Oberfläche mit `startx`.
Die `respawn`-Angabe in der `inittab` sorgt übrigens dafür, dass der angegebene Befehl wiederholt wird, sobald das entsprechende Terminal (durch Ausloggen) geschlossen wird.

✎ Finde heraus, welche Init-Skripte im Default-Runlevel Deines Rechners beim Booten *nicht* gestartet werden.

## 10 Der Ausschaltknopf

Bei einem Multiuser-Multitasking-Betriebssystem ist der Ausschaltknopf als Allheilmittel für Rechnerprobleme nicht so recht am Platz, denn was kann eine unschuldig remote eingeloggte, Mail-lesende Benutzerin dafür, dass bei mir gerade die Textverarbeitung oder der lokale X-Server klemmt?
Aber nicht nur auf eventuelle andere Nutzerinnen gilt es Rücksicht zu nehmen, sondern vor allem auf (zum Beispiel über Initskripte gestartete) Prozesse, die im Hintergrund den verschiedensten Aufgaben nachgehen. Einfach ausschalten lässt sie möglicherweise mit gerade zum Schreiben geöffneten Dateien einen gewaltsamen Tod sterben – die Folge sind Inkonsistenzen im Dateisystem und Datenverlust.
Ausschalten sollte daher der Notfall bleiben – im Normalfall fährt frau ihren Rechner geordnet runter. Dabei bekommen alle Prozesse ein Signal von `init` zugesandt, dass sie schleunigst mit den aktuellen Schreiboperationen fertig werden und sich beenden sollen.
Herunterfahren heißt dabei nichts anderes als in den Runlevel 0 oder 6 wechseln. Außer mit `telinit` (meist übrigens nur ein anderer Name, ein *Link*, für `init`) geht das auch mit den Befehlen `shutdown`, `reboot` und `halt`.
Alle drei sind in der Regel nur der Superuserin zugänglich. `halt` fährt den Rechner ohne Umschweife herunter; wenn der Rechner steht, ist es Zeit, den Ausschaltknopf zu betätigen, wenn das Motherboard nicht selbst dafür sorgt.

[33] Nur der Vollständigkeit halber: 38400 gibt die Baud-Rate, also die Anzahl der Symbole des übertragenen Signals pro Sekunde an. `tty1` und tty2 das zu benutzende Device-File unterhalb des Verzeichnisses `dev` (vgl. Seite 4.3)

Beim Sys-V-Init gibt es zudem Verzeichnisse namens `rc`*Runlevel*`.d`. Dort liegen Verweise (*Links*) auf besagte Init-Skripte. Sie definieren, welche Skripte tatsächlich ausgeführt werden: Beginnt der Name des Links mit einem S, wird das verlinkte Skript beim Wechsel in diesen Runlevel (z. B. beim Hochfahren des Rechners) mit dem Parameter `start` aufgerufen. Ist der erste Buchstabe ein K (wie „kill“), ruft `init` es beim Wechsel in einen anderen Runlevel (beispielsweise 1 oder 6 zum Herunterfahren) mit dem Parameter `stop` auf.
Die verschiedenen Skripte werden jeweils in der Reihenfolge der auf den ersten Buchstaben folgenden Zahlen abgearbeitet.
Zum Glück halten sich im Zuge der Umsetzung der *Linux Standard Base* (LSB) nach Jahren der Uneinigkeit immer mehr Distributoren (wieder) an dieses einfache Konzept. Bei nicht mehr ganz taufrischen Distributionen gilt es jedoch aufzupassen: So reicht es bei ältlichen SuSE-Ausgaben nicht, dass Startlinks vorhanden sind – um einen Dienst zu starten, muss zusätzlich auch noch eine Variable in der Datei `/etc/rc.config` gesetzt werden[32]. Auch wenn sich die LSB-Vorgabe `/etc/init.d` als Container-Verzeichnis für die Init-Skripte inzwischen weitgehend durchgesetzt hat, sollte frau im Zweifelsfall lieber auch `/sbin` (alte SuSE-Distributionen), `/etc` oder `/etc/rc.d` durchsuchen.

✎ Wo liegen die Initskripte und die jeweiligen Links auf Deinem Rechner?

✎ Welches Initskript ist dafür verantwortlich, dass der Cron-Daemon startet?

Je nach Runlevel startet `init` nun verschiedene Prozesse: Er sieht in der Datei `/etc/fstab` nach, welche Partitionen er außer der Root-Partition noch einbinden, sprich: *mounten*, muss. Bevor er das tut, überprüft er das darauf enthaltene Dateisystem (so möglich) auf Konsistenz und repariert es ggf.
Da das beim *ex2fs*-Dateisystem sehr lange dauern kann, wird ein solcher *Filesystemcheck* mit dem Programm `fsck` nur dann durchgeführt, wenn eine bestimmte (mit dem Befehl `tune2fs` einstellbare) Anzahl Bootvorgänge überschritten wurde oder wenn der Rechner nicht ordnungsgemäß heruntergefahren wurde (vgl. Abschnitt 10 ab Seite 51). Frau sollte lediglich im Kopf behalten, dass das – in der Vergangenheit standardmäßig benutzte – Ext2fs mittlerweile nicht mehr konkurrenzlos dasteht, sondern immer öfter andere Dateisysteme zum Einsatz kommen können, die ihre eigenen Spezialtools mitbringen.
Auch das Root-Filesystem wird geprüft. Damit bis zu diesem Zeitpunkt nicht versehentlich etwas auf ein möglicherweise inkonsistentes Filesystem geschrieben wird, ist es bis nach der Überprüfung nur lesbar (*read-only*) gemountet und wird anschließend *read-write* remountet.
Bei manchen Linux-Distributionen sorgt das (distributionsabhängige) Boot-Skript `rc.boot` dafür, dass ein benutzbares System hochfährt, auf dem allerdings noch kaum ein Dienst läuft. In neueren Distributionen sollten die entsprechenden Aufrufe allerdings in jeweils eigene Init-Skripte ausgelagert werden, die ins Verzeichnis `/etc/rcS.d` verlinkt werden.
Dann werden nacheinander die Dienste gestartet, die für das jeweilige Runlevel vorgesehen sind; zum Schluss kommt manchmal noch ein `rc.local`-Skript zum Tragen, in dem die lokale System-

[32] Dieses Konzept war übrigens FreeBSD, einem anderen freien Unix, abgeschaut.





### 5.3 Besitzerinnen ändern

Vor allem `root` benötigt öfters eine Möglichkeit, eine Datei einer anderen Besitzerin zu übertragen. Dazu gibt es zwei Wege:

- Besitzerin und Gruppe gleichzeitig ändern:

  ```
  chown neueUserin.neueGruppe dateiname
  ```

- Besitzerin und Gruppe extra ändern:

  ```
  chown neueUserin dateiname
  chgrp neueGruppe dateiname
  ```

✎ `chmod`, `chgrp` und `chown` lassen sich auch auf alle Dateien und Unterverzeichnisse in einem Ordner gleichzeitig anwenden. Um welche Option muss frau den Befehl dann jeweils erweitern? Nutze diese Option, um jeweils allen anderen das Leserecht auf alle Dateien eines Verzeichnisses zu entziehen!

## 6 Weitere Befehle zur Dateiverwaltung

### 6.1 Dateien und Verzeichnisse löschen

Dateien werden mit dem Befehl `rm Dateiname` gelöscht. Aber Vorsicht! Bei Linux ist weg auch wirklich weg! Es kann zu ganz bösen Fehlern kommen, wenn sich frau aus Versehen verschreibt. Eine der ärgerlichsten Fehleingaben lautet:

```
rm nichtwichtig. *
```

Der Stern ist eine sogenannte *Wildcard* und steht für „beliebige Anzahl aller möglichen Zeichen“. Problematisch ist das Leerzeichen zwischen Dateinamen und Stern. Damit verschwinden nicht etwa nur Dateien wie `nichtwichtig.gif` und `nichtwichtig.jpg`, sondern alle Files im aktuellen Verzeichnis: Der Stern passt auf alle Dateinamen. Dass nebenbei auch noch eine eventuell vorhandenen Datei `nichtwichtig`. dran glauben muss und mit einem Punkt beginnende Dateien erhalten bleiben, fällt da nicht mehr ins Gewicht.

✎ Lege ein Verzeichnis namens `loeschtest` an und wechsele hinein. Füll es mit den Dateien `nichtwichtig.txt`, `nichtwichtig`, `blafasel`, `.bla` und dem Verzeichnis `ordner`. Überprüfe die obengenannten Aussagenmit dem Befehl `rm nichtwichtig. *`! Was passiert mit `ordner`? Warum?

Kommt die Option `-r` (für „rekursiv“) zum Einsatz, lassen sich mit `rm` auch Verzeichnisse (samt Inhalt) löschen.

✎ Probiere das rekursive Löschen am Verzeichnis `loeschtest` aus der vorherigen Aufgabe aus! Ist das Unterverzeichnis `loeschtest/ordner` anschließend immer noch vorhanden?

Vorsichtige Naturen setzen auf die Option `-i`, die `rm` durch eine Nachfrage entschärft. Sind sie zudem noch faul, setzen sie sich einen *Alias*, der `rm` durch `rm -i` „ersetzt“:

```
alias rm="rm -i"
```

Damit muss jede Löschaktion zusätzlich („interaktiv“) bestätigt werden. Böse Überraschungen bleiben so aus. Um trotzdem ungenervt einmal ein ganzes Unterverzeichnis löschen zu können, unterbindet die Option `-f` für „force“ die Fragerei, und alles wird auf einmal vernichtet.

**Alias** Ein Alias ist ein Spitzname. Damit lassen sich lange Kommandos abkürzen oder ungenügende umschreiben. Ein Alias kann entweder in der Shell gesetzt werden – dann gilt er nur so lange, wie diese Shell läuft – oder frau „verewigt“ ihn in einer der persönlichen Konfigurationsdateien `.bashrc` oder `.profile` geschrieben. Dann gilt er überall.
Nutzerinnen des besagten `rm`-Aliases sollten doppelt aufpassen, wenn sie an eine fremde Maschine kommen: Wenn es diesen Alias dort nicht gibt, kann das fatale Folgen haben...

✎ Schau Dir mit dem Kommando `alias` an, welche Aliase Du derzeit benutzen kannst!

### 6.2 Dateien und Verzeichnisse kopieren oder umbenennen

Zum Kopieren dient der Befehl

```
cp name1 name2
```

Ohne Angabe von Optionen kopiert cp[18] nur Dateien. Mit der Option `-r` (für „rekursiv“) dupliziert es auch ganze Verzeichnisse mitsamt aller darin enthaltenen Unterordner.
Zum Umbenennen dient der Befehl

```
mv name1 name2
```

Er funktioniert sowohl bei Dateien als auch bei Verzeichnissen, bei letzteren allerdings nur dann immer, wenn sie sich auf der selben Partition befinden.

### 6.3 Verzeichnisse manipulieren

Ein leeres Verzeichnis legt der Befehl `mkdir verzeichnisname` an. `rmdir` löscht es wieder – allerdings nur, solange es keine Dateien enthält.

✎ Lege ein Verzeichnis `temporaer` an (Du kannst es gern mit einigen anderweitig herkopierten Dateien füllen). Wieviele Kopien davon gibt es nach den drei Befehlen `cp temporaer TMP`, `cp -r temporaer temp` und `mv temporaer tmp`? Warum?

Möchte frau ein neues Verzeichnis und darin gleich noch eins anlegen, kann sie dies in einem Rutsch mit der `mkdir`-Option `-p` („parent“) erledigen:

```
mkdir -p verzeichnis/unterordner
```

✎ Gibt es die Option `-p` auch für `rmdir`? Wenn ja: Kann sie als Ersatz für `rm -r` dienen?

### 6.4 Dateien suchen

Manchmal weiß frau einfach nicht mehr, wo sie eine Datei abgelegt hat. Da hilft der Befehl

```
locate Suchstring
```

[18] zumindest in der GNU-Implementation





Hat es der Bootloader geschafft, den Kernel zu finden und damit begonnen, ihn in den Hauptspeicher zu laden, ist seine Aufgabe erledigt. Das Betriebssystem ist nun soweit im Speicher, dass es sich und seine Treiber „an den Haaren herbeiziehen“ und die Prozesstabelle aufbauen kann.

✎ Sieh Dir das Boot-Protokoll des Kernels nachträglich mit dem Befehl `dmesg` an! Was ist dabei zum Beispiel passiert?

Allerdings kann frau mit dem Rechner zu diesem Zeitpunkt immer noch nichts anfangen. Zunächst muss ein Prozess Nummer 1 namens `init` eine Menge Arbeit verrichten. Er hat momentan nur die Dateien und Programme zur Verfügung, die auf der *Root-Partition* zu finden sind. Das sollten mindestens die Verzeichnisse `/sbin` (Programme, die i. A. nur von der Superuserin benötigt werden), `/lib` (Bibliotheken, die von den Programmen in `/sbin` und `/bin` geladen werden), `/etc` (Konfigurationsdateien) und `/bin` (Programme, die auch für normale User von Interesse sein können) sein.
Welche Aufgaben anstehen, findet `init` in der Datei `/etc/inittab` beschrieben. Zunächst muss die „Mutter aller Prozesse“ wissen, in welchen Betriebszustand, in welchen *Runlevel*, sie schalten soll. Wenn der Kernel beim Laden keine andere Option per „Kommandozeilenparameter“ mitgeteilt bekam, ist das der *Default-Runlevel*, der in der `inittab` niedergelegt ist.

✎ Finde in der `/etc/inittab` heraus, in welchen Runlevel Dein Rechner von sich aus startet. Welche Zeile könnte das sein?

**Runlevel:** Ein Unixrechner kann ohne Weiteres zur Laufzeit aus einer unvernetzten Workstation zu einem Internetserver werden. Einzige Voraussetzung ist, dass entsprechende Betriebszustände definiert sind. Wie diese Runlevel aussehen und welche Nummern sie tragen, unterscheidet sich von Distribution zu Distribution. Gemeinsam sind ihnen lediglich die Runlevel 0 (*halt*), 6 (*reboot*) und der *Single-User-Mode* 1 (*Single-User-Mode*: Bei letzterem handelt es sich um einen Wartungsmodus, in den die Superuserin zum Reparieren gehen kann. Hier gibt es (meist) kein Netzwerk, keine Dienste und nur das, was auf dem Root-Filesystem zu finden ist. Der Rechner lässt sich also zum Rebooten bringen, indem `root` einfach den Runlevel wechselt: `telinit 6`
Unter den Runleveln 2–5 gibt es meist einen, der den X-Server startet; ansonsten ist ihnen gemeinsam, dass sie Multiuser-Betrieb ermöglichen. Immerhin dokumentieren die meisten Distributoren in der `/etc/inittab`, welche groben Eigenschaften die verschiedenen Betriebszustände haben.
Traditionell gibt es unter Unix zwei *Init-Systeme*: *Simple Init* und *System-V-Init*. Da außer Exoten wie Slackware kaum eine Distribution auf ersteres setzt, beschäftigen wir uns hier nur mit letzterem.
In einem Verzeichnis namens `init.d` liegen verschiedene *Start-Stopp-Skripte*, die den Rechner softwareseitig netzwerkfähig machen und verschiedene Dienste konfigurieren und starten: pro Dienst ein Skript. Ruft frau sie mit dem Parameter `start` auf, startet der entsprechende Service, mit dem Parameter `stop` wird er beendet. Auch andere Parameter wie `restart` sind möglich, spielen beim Booten und Herunterfahren des Rechners allerdings keine Rolle, sondern erlauben es der Systemadministratorin, Dienste bei Konfigurationsänderungen schnell neu zu starten etc.

hinzu, dass `root` diese Datei von Hand bearbeiten muss – der `crontab`-Befehl übernimmt dies nicht. In dieser Datei wird meist ein *run-parts*-Programm aufgerufen, das Skripte abarbeitet, die in den Verzeichnissen `/etc/cron.daily` bis `/etc/cron.monthly`[31] abgelegt sind.

✎ Wann wird der folgende Eintrag abgearbeitet?

```
0 13 * 9 5-7 echo ‘Der Kurs geht weiter¡
```

✎ Lass Dir in 10 Minuten von Cron eine Übersicht über die Plattenbelegung (`df`) schicken. Die aktuelle Zeit findest Du mit dem Kommando `date` heraus.

## 9 Bootvorgang

Wer sich jetzt fragt, wie es denn kommt, dass da im Hintergrund dieser Cron-Daemon überhaupt werkeln kann, bekommt die Antwort an dieser Stelle: Er muss natürlich irgendwann gestartet worden sein, und zwar automatisch beim Booten. Selbstverständlich hat eine Linux-Administratorin es in der Hand, diesen Automatismus zu beeinflussen. Wie das geht, wird deutlich, wenn frau sich anschaut, was beim Booten passiert.
Wenn das BIOS seine Überprüfungen beendet hat, lädt es den Startcode des Betriebssystems. Auf Linuxrechnern – egal, ob nur Linux oder auch andere Betriebssysteme installiert sind – ist beim Booten von Festplatte an dieser Stelle in der Regel ein im Master-Boot-Record der Festplatte installierter *Bootmanager* gefragt, der es erlaubt, zwischen mehreren Operativsystemen zu wählen (das können auch verschiedene Linuxinstallationen sein, vgl. Seite 9). Er startet das passende OS, d. h., er lädt den Kernel.
Der unter Linux klassischerweise eingesetzte Bootmanager heißt *LILO* („Linux Loader“). Mit *Grub* gibt es aber eine – zunehmend verwendete – Alternative.

✎ Schau in die `/etc/lilo.conf` und versuche herauszufinden, welche Bootmöglichkeiten Dein Rechner bietet.

**Woher einen Kernel nehmen? Das Bootmanager-Dilemma:** Wenn der Bootmanager den Kernel laden will, befindet sich noch kein Betriebssystem und damit auch keine Unterstützung für Dateisysteme im Hauptspeicher. Der Linuxkernel ist zwar, wie wir im Kapitel 2.1 ab Seite 7 gesehen haben, nichts weiter als eine Binärdatei, aber das Wissen um `/boot/vmlinuz` o. ä. nutzt uns mangels Dateisystem zu diesem Zeitpunkt herzlich wenig. Der Bootloader muss also wissen, an welcher physikalischen Stelle auf der Festplatte sich der Kernel befindet.
Deshalb reicht es beim LILO nicht, die Konfigurationsdatei `/etc/lilo.conf` zu editieren (per Hand oder per Installationstool), denn er kann beim Booten nicht in diese Datei schauen. Daher schreibt frau mit dem Kommando `/sbin/lilo` in den Bootsektor, wo auf der Festplatte ein (hoffentlich startbarer) Kernel liegt. Anders ausgedrückt wird der Bootmanager damit überhaupt erst installiert.
Es gilt also, zum einen zwischen dem Bootloader LILO und dem Kommando `lilo` zu unterscheiden, das den Bootmanager erstellt und schreibt. Zum anderen reicht es nicht, einen Kernel zu kompilieren. Damit er gestartet werden kann, muss er in die Lilo-Konfigurationsdatei als neue Bootmöglichkeit eingetragen und anschließend `/sbin/lilo` aufgerufen werden. Solange frau noch nicht sicher ist, dass der neue Kernel auch ordentlich bootet und funktioniert, sollte sie den Eintrag für den bislang geladenen Kernel noch nicht entfernen, sondern ihn als alternative Bootmöglichkeit anbieten.

[31] Nicht (vollständig) LSB-kompatible Distributionen verwenden hier oft andere, jedoch meist ähnliche Namen.





Das geht rasend schnell, da `locate` auf die Datenbank `locatedb` zugreift, die meist automatisch erstellt wird. Wenn nicht, kann `root` unter Linux in aller Regel mit dem Befehl `updatedb` nachhelfen.
Im Übrigen findet `locate` auch dann nicht alles: Gerade in Rechenzentren schließen viele Administrator(inn)en zum Beispiel die Home-Verzeichnisse der User beim Erzeugen der `locate`-Datenbank aus, damit diese nicht zuviel Einblick in fremde Verzeichnisse bekommen.
Alternativ bittet die Benutzerin den Befehl `find` zu Hilfe. Seine Syntax lautet

```
find Startverzeichnis -name gesuchteDateiOderMuster
```

Mit der (am Ende angefügten) Option `-exec Befehl {} \;` führt er für jede gefundene Datei den gewünschten Befehl aus: `{}` steht dabei als Platzhalter für das Suchergebnis. Das Semikolon ist nötig, um die so generierten Kommandos voneinander zu trennen. Damit die Shell es nicht vorzeitig als eigenes Sonderzeichen interpretiert, muss es mit dem Backslash geschützt werden.
Wer Lust auf ein Kommando mit richtig vielen komplexen Optionen hat, wird in der Manpage zu `find` fündig...

✎ Finde das KDE-Startskript `startkde` mit der Hilfe von `locate`!

✎ Wechsle ins Verzeichnis `/tmp` und suche mit `find` nach dem in der Aufgabe auf Seite 32 erstellten `temp`-Directory (oder alternativ nach einer anderen Datei irgendwo unterhalb Deines Homeverzeichnisses).

✎ In welchen Verzeichnissen liegt der „Quellcode“ der verschiedenen Manpages zu `man`? (vgl. Aufgabe auf Seite 18)

### 6.5 Verschiedene Arten, eine Datei anzuschauen

Selbstverständlich kann frau vom Webbrowser über Textverarbeitungen bis hin zu Texteditoren alle möglichen Programme bemühen, um sich den Inhalt von menschenlesbaren Textdateien anzuzeigen. Aber wozu mit Kanonen auf Spatzen schießen? Mit einer Reihe von Kommandozeilentools kommt sie oft viel schneller zum Ziel:

`cat` („concatenate“) zeigt Dateiinhalte „ungebremst“ an.

`more` ist ein einfaches Anzeigeprogramm, das sich automatisch beendet, wenn das Ende der Datei erreicht ist.

`less` Dieser mächtige Nachfolger von `more` kennt u. a. folgende Tasten-Kommandos:

- (Leer) blättert eine Seite vor,
- (↵) blättert eine Zeile vor,
- (B) blättert eine Seite zurück,
- (H) Hilfe zu den in `less` zur Verfügung stehenden Kommandos,
- (Q) beendet `less`,
- `/Suchmuster` sucht im Text nach *Suchmuster*, springt an diese Stelle und markiert es. Weiter mit (N).

Auf vielen Linux-Systemen existieren zudem die Programme `zcat`/`zless` und `bzcat`/`bzless`, mit denen sich mit `gzip` bzw. `bzip2`[19] komprimierte Textdateien anzeigen lassen, ohne sie vorab händisch entpacken zu müssen.

[19] vgl. Seite 44

### 6.6 Extrawurst: die Drucker

Drucken unter Linux ist ein sehr komplexes und im Wandel befindliches Thema, das den Rahmen dieses Kurses sprengen würde. Deswegen ignorieren wir einmal die Art und Weise, wie aus einer Datei über das Zwischenformat PostScript ein bedrucktes Blatt Papier wird und beschränken uns auf das Abschicken und Verwalten von Druckjobs auf der Kommandozeile.
Ein lokaler Drucker „hängt“ in der Regel an der parallelen Schnittstelle, die unter Linux als `/dev/lp1` im Verzeichnisbaum ansprechbar ist. Da Linux sie als Datei verwaltet, könnte das Dokument einfach an den Drucker kopiert werden:

```
cp datei /dev/lp1
cat datei > /dev/lp1
```

Normalerweise hat aber nur `root` Zugriff auf Gerätedateien. Zudem will nicht jede Benutzerin wissen müssen, an welchem Device der Drucker hängt, und außerdem bietet diese direkte Druckeransprache keinen Schutz davor, dass mehrere Druckaufträge gleichzeitig versuchen, auf Papier zu kommen. Aus all diesem Gründen nimmt frau die Dienste des Druckerdaemonen `lpd` oder seines moderneren Nachfolgers `cupsd` des „Common Unix Printing System“ (CUPS) in Anspruch. Wenn der Drucker gar ein Netzwerkdrucker mit eigener IP-Adresse ist, bleibt ohnehin nichts weiter übrig ...
Der Drucker-Daemon stellt alle ankommenden Druckjobs in eine Reihe (Queue) und arbeitet sie hintereinander ab, sodass Konflikte vermieden werden (*Scheduler*).

**Abschicken eines Druckjobs** Sollen Druckjobs an den Drucker-Daemon übermittelt werden, kommt das Kommando

```
lpr dateiname
```

zum Einsatz, wenn es nur einen einzigen Drucker im System gibt oder der Standarddrucker in der Umgebungsvariablen `PRINTER` eingetragen ist. Können mehrere Drucker angesprochen werden, so muss der gewünschte mit der Option `-P druckername` angewählt werden.
Welche Drucker es gibt, steht in der Datei `/etc/printcap`, die bei CUPS-basierten Systemen praktisch nur die Druckernamen enthält:

```
davinci:
```

Dieser Drucker ließe sich also mit `lpr -Pdavinci datei` ansprechen.
Wem das zu umständlich ist: Der KDE-Druckdialog lässt sich auf der grafischen Oberfläche jederzeit mit dem Kommandozeilenbefehl `kprinter druckdatei` aufrufen...

**Auflisten aktueller Jobs** `lpq` zeigt, welche Druckaufträge gerade abgearbeitet werden. Dieses Kommando liefert außerdem die Information, wer den Job abgeschickt hat, welche Datei gedruckt wird und wie groß sie ist:

```
$ lpq
lp1 is ready and printing
Rank    Owner    Job   Files                      Total Size
active  bille    395   /home/bille/inf/linux.ps   562130 bytes
1st     bille    396   /home/bille/zug.ps         86626  bytes
```

Auch hier lässt sich über die `-P`-Option ein bestimmter Drucker spezifizieren.





✎ Starte `mozilla` im Vordergrund einer Shell. Schick den Webbrowser manuell in den Hintergrund. Starte das Programm `xeyes` in derselben Shell im Hintergrund. Hole `mozilla` wieder in den Vordergrund. Beende Mozilla, ohne auf sein Menü zuzugreifen. Suche nach dem `xeyes`-Prozess in der Prozesstabelle und beende ihn mit dem `kill`-Befehl.

Neben diesen traditionellen Unixtools gibt es auf vielen Linuxsystemen noch zwei Werkzeuge, die das Verwalten von Prozessen leichter machen: `pstree` zeigt in einer symbolischen Baumstruktur an, welcher Prozess von welchem aufgerufen wurde, sprich, welcher Prozess eine *Tochter* von welchem *Elternprozess* ist. In dieser Darstellung lässt sich gut nachvollziehen, dass `init` die Mutter aller Prozesse ist.

✎ Schau Dir mit `pstree` den Prozessbaum an und probiere die Wirkung einiger Optionen, die Du in der Manpage findest.

Das Heraussuchen von PIDs ist eine recht langwierige Angelegenheit. Schneller ginge es, wenn frau statt der Nummer einfach den Namen des Prozesses angeben könnte. Das geht zwar nicht immer, aber dafür werden von `killall` oder als letzte Rettung auch `killall -9` gleich alle Prozesse gleichen Namens mit in den Abgrund gerissen.

✎ Starte drei `xeyes` und beende sie alle gleichzeitig.

### 8.4 Die Zeit unter Kontrolle

Nicht immer will frau am Rechner hocken, wenn der eine bestimmte Aufgabe ausführen soll. Zu diesem Zweck gibt es mit `cron(d)` einen Daemon, der im Hintergrund nichts anderes tut, als nachzuschauen, ob in der aktuellen Minute ein Job für ihn ansteht.
Aufträge für Cron werden in *Crontabellen* eingetragen. Neben der systemweiten `/etc/crontab` gibt es auch für jede Userin eine. Um diese anzulegen oder zu ändern, verwendet frau das Kommando `crontab -e` („edit“). Sofern die Variable `EDITOR` oder `VISUAL` nichts anderes sagt, startete jetzt der `vi`.
Der Job muss in einer ganz bestimmten Form in die Crontabelle eingetragen werden:

```
Minute(n) Stunde(n) Tag(e) Monat(e) Wochentag(e) Kommando
```

Leerzeichen dienen als Spaltentrenner; wenn ein Job zweimal am Tag um 12 und 24 Uhr ausgeführt werden soll, kann frau dies mit Komma (in diesem Fall `0,12` in der Stundenspalte) angegeben. Soll ein Job jeden Tag von Montag bis Freitag ausgeführt werden, schreibt frau `1-5` in die Wochentagsspalte. Ein `*` in einer Zeitspalte bedeutet, dass die jeweilige Spalte egal ist.
Als Kommando schreibt frau die Kommandozeile, die sie auch in der Shell eingeben würde. Aber Achtung: Manche Programme reagieren nicht wie gewünscht, wenn sie ohne Terminal vom Cron-Daemon aufgerufen werden. Ehe frau sich also vom Rechner entfernt, sollte sie überprüft haben, dass alles glatt geht. Wichtig ist zudem, darauf zu achten, dass in einer Jobzeile keine Zeilenumbrüche vorkommen. Sonst meckert `crontab` beim Abspeichern.
X-Clients in Cronjobs aufzurufen ist übrigens ein Problem: Wenn die jeweilige Userin nicht ohnehin grafisch eingeloggt ist, geht das schief.
Grafische Frontends wie `kcron` können dabei helfen, die ersten Hürden bei der gut in der Manpage zu `crontab(5)` erklärten Syntax zu überwinden.
Wenn Cron einen Job ausführt, schickt er (so nicht anders angegeben) die Ausgabe als Mail an die Auftraggeberin. Schon deshalb lohnt es sich, einen funktionierenden lokalen Mailserver aufzusetzen.
Die systemweite Aufgabensammeldatei `/etc/crontab` enthält vor dem Kommando noch eine zusätzliche Spalte: die Userin, in deren Namen der Auftrag ausgeführt wird. Erschwerend kommt

benötigt, wäre es gut, wenn frau gleichzeitig noch etwas anderes tun könnte. Bei über Menüs oder Icons aufgerufenen Kommandos ist das ohnehin der Fall[28], doch wie bekommt frau eine Shell wieder frei?
Hier hilft Unix durch seine Mehrprozessfähigkeit (Stichwort: Multitasking): Bekommt ein Kommando ein `&` anhängt, wird es im Hintergrund gestartet, und die Shell meldet sich sofort wieder mit dem Prompt zurück. Angenommen, wir haben ein Programm mit dem Namen `coffee`, dessen Ausführung viel Zeit in Anspruch nimmt, und eines namens `tea`, das deutlich schneller geht. Dann würden wir folgendermaßen verfahren[29]:

```
coffee &    # Programm coffee wird im Hintergrund abgearbeitet
[1] 26070
tea         # tea  wird normal gestartet
```

Wenn sich das Programm `coffee` beendet, erhält die Benutzerin eine Nachricht der Art

```
[1]   Done                    coffee
```

in der aufrufenden Shell. Die Zahl in eckigen Klammern bezeichnet die Jobnummer dieses Hintergrundprozesses. Beim Start erfährt frau zudem seine *Prozessidentifikationsnummer* (PID, im Beispiel 26070).
Wer nicht mehr so genau weiß, welche Programme in der aktuellen Shell noch im Hintergrund laufen, kann sie sich mit dem Kommando `jobs` ansehen. Es zeigt den Status und die Jobnummer der in dieser Shell laufenden Hintergrundprozesse an.
Hat frau vergessen, einen Prozess beim Aufruf in den Hintergrund zu schicken, ist das keine Katastrophe. (Strg)+(z) hält ihn an; anschließend „verschiebt“ ihn das Kommando `bg` („background“) in den Hintergrund. Ebenso lässt sich der letzte Hintergrundprozess mit `fg` („foreground“) wieder in den Vordergrund holen. Laufen mehrere Prozesse im Hintergrund, gibt frau `fg` die passende Jobnummer als Option mit auf den Weg.
Läuft der Prozess im Vordergrund (wurde also ohne das `&` gestartet), so lässt er sich normalerweise (leider nicht immer) mit (Strg)+(c) abbrechen. Diese Tastenkombination signalisiert dem Prozess, einen *geordneten Abbruch* durchzuführen. Das bedeutet zum Beispiel, daß geöffnete Dateien noch geschlossen werden u. a.
Wenn ein Prozess nicht mehr auf (Strg)+(c) reagiert, ist das dennoch kein Grund zur Panik: Wir arbeiten schließlich mit einem Multitasking-System. Frau geht einfach in ein neues Fenster oder an ein anderes Terminal und sucht die Prozessidentifikationsnummer des hartnäckigen Prozesses. Dazu verwendet sie das Kommando `ps` („process status“).
`ps` ohne weitere Zusätze listet die in der aktuellen Shell laufenden Prozesse tabellarisch auf. Um eine ausführliche Liste aller auf der Maschine gestarteten Prozesse zu bekommen, verwendet frau unter Linux die Optionen `aux`; auf anderen (System-V-)Unixsystemen benötigt sie dagegen die Optionen `-ef`[30], denn `ps` gehört zu den traditionellen Unixtools, die zwar überall vorhanden sind, sich aber leider überall ein wenig anders verhalten, und das mittlerweile sogar auf unterschiedlichen Linux-Distributionen. Hier empfiehlt sich ein Blick auf die Manpage.
In der `ps`-Ausgabe sucht frau in der Spalte `COMMAND` bzw. `CMD` den Prozess, den sie *abschießen* möchte. In der Spalte `PID` findet sie die zugehörige Prozessidentifikationsnummer. Jetzt kann sie den hängengebliebenen Prozess durch den Befehl `kill` *pid* abbrechen. Führt das immer noch nicht zum Erfolg, ruft sie `kill` mit der Option `-9` auf – kein Prozess kann dieses Signal ignorieren.

[28] Und wenn nicht, ist das ein Bug...
[29] Das Hash-Zeichen `#` leitet in der Bash einen Kommentar ein; darauffolgende Zeichen ignoriert sie. Damit könnten wir die jeweilige Erklärung problemlos mit abtippen, wenn uns danach wäre ...
[30] ... die übrigens auch unter Linux funktionieren





**Löschen von Druckjobs** Jeder Job bekommt eine fortlaufende Nummer (mit `lpq` sichtbar). Über diese kann er mit `lprm` „abgeschossen“ werden, sofern er noch nicht im Druckerspeicher gelandet ist:

```
$ lprm 396
dfA10ratschni dequeued
```

✎ Wie würdest Du den Druckjob Nr. 10 auf dem Drucker `davinci` aus der Queue entfernen?

auf. Läuft das Skript nicht durch, ist meist Nachinstallation von Third-Party-Tools und -Bibliotheken angesagt.
Das `configure`-Skript legt u. a. auch fest, wohin welche Dateien letzten Endes installiert werden sollen. In der Regel ist das `/usr/local`. Wer Wert darauf legt, die selbstkompilierte Software später auch wieder rückstandslos zu deinstallieren, wird jedoch eher ein eigenes Verzeichnis wie `/usr/local/extra-verzeichnis` wählen[26]. Das geht (meist) mit der `configure`-Option `-prefix`:

```
./configure --prefix=/usr/local/extra-verzeichnis
```

`configure` kennt meist noch weitere Optionen, die z. B. die Funktionalität des Binaries anpassen. Welche das sind, erfährt frau mit `./configure -help`.
Lief `configure` problemlos durch, startet `make` den Kompiliervorgang. Für die endgültige Installation braucht frau `root`-Rechte, sofern sie nicht in ein eigenes Verzeichnis installiert: `make install` kopiert dann alle relevanten Daten an Ort und Stelle. Alternativ baut das Tool `checkinstall` ein (intermediäres) `.rpm`- oder `.deb`-Paket und installiert es, sodass zur Deinstallation der Paketmanager zum Zuge kommen kann[27].
Wer erst einmal wissen möchte, was `make install` vorhat, lässt den Befehl zunächst ohne `root`-Rechte laufen: Das gibt Fehlermeldungen, wenn das Makefile Dateien in Verzeichnisse kopieren will, in denen eine unprivilegierte Nutzerin keine Schreibrechte hat. Sinnvoller ist zwar die Trockenübung `make -n install`, die nur so tut als ob `make install` ausgeführt werden soll, aber speziell bei automatisch generierten Makefiles ist die Ausgabe nicht sehr allgemeinverständlich.
Natürlich ist frau nicht vor Kompilierfehlern gefeit: Was sich auf einem System problemlos kompilieren lässt, kann auf einem anderen Probleme machen. Wer sich in C oder C++ auskennt, kann hier sicher die eine oder andere Sache ausbügeln, doch bei komplexer Software ist das oft nicht so einfach. Dann bleibt immer noch die Möglichkeit, eine andere Version auszuprobieren, im Netz nach Leidensgenoss(inn)en und deren Problemlösungen zu fahnden oder aber den Autor(inn)en der Software eine Mail zu schreiben. Viele reagieren darauf ausgesprochen hilfsbereit, sofern frau soviel wie möglich relevante Informationen (welche Distribution, welche Compilerversion, welche Version benötigter Bibliotheken, welche Fehlermeldung etc.) mitliefert. Bei großen Projekten gibt es Bug-Tracking-Systeme, in die diese Informationen eingetragen werden.

✎ Lade das Paket `stow`-Paket http://ftp.debian.org/debian/pool/main/s/stow/stow_1.3.3.orig.tar.gz herunter und installiere es in Deinem Homeverzeichnis!

✎ Welche Version des C-Compilers `gcc` käme auf dem Pool-Rechner zum Einsatz? Finde den passenden Aufruf mit der `gcc`-Option `-help` heraus!

## 8.3 Prozesse im Griff

Dass ein Rechner etwas „tut“, verdanken wir Prozessen. Bei ihnen handelt es sich um eine Folge von Aktivitäten, die zwischen dem Aufruf eines Programms und dessen Ende stattfinden. Wer sein System unter Kontrolle haben möchte, muss herausfinden können, welche Prozesse laufen, wie sich widerspenstige Programme zum Aufgeben zwingen oder dazu bringen lassen, die Kommandozeile nicht zu blockieren.
Von jedem Kommando – sei es ein auf der Kommandozeile eingetipptes oder ein durch die Anwahl eines Menüpunkts aufgerufenes – wird ein neuer Prozess gestartet. Manchmal dauert die Ausführung eines Programms jedoch sehr lange. Sofern dieses Programm keine Benutzereingaben

[26] ... und die installierten Dateien eventuell von Hand oder mit einem Tool wie `stow` so verlinken, dass sich der Suchpfad nicht ins Unendliche aufbläst.
[27] In diesem Fall ist das Setzen eines alternativen Präfixes beim `configure`-Lauf meist nicht nötig.

Paket, ist der gesamte Dateiname gefragt (`wwwoffle-2.6b-2.i386.rpm`).
So ließe sich das besagte `wwwoffle`-Paket mit

```
rpm -iv wwwoffle-2.6b-2.i386.rpm
```

installieren, wobei das Flag `-v` für „verbose“, also gesprächige Ausgaben sorgt. Will frau zudem einen „Fortschrittsbalken“ aus `#`-Zeichen sehen, fügt sie die Option `-h` („hash“) mit an.
Sofern das Paket bereits installiert ist, bekommt frau eine entsprechende Fehlermeldung. War ein Update auf eine neue Version das ursprüngliche Ansinnen, so benutzt sie statt `-i` das Flag `-U` („update“ statt „install“).
Wieder los wird frau ein installiertes Paket mit der Option `-e` wie „erase“:

```
rpm -e wwwoffle-2.6b
```

Zum Einholen von Informationen über Pakete dient die Option `-q` („query“). Geht es dabei um noch nicht installierte Pakete, so kommt noch ein `-p` („package“) dazu.
Will frau wissen, welche Pakete auf ihrem Rechner installiert sind, findet sie das mit `rpm -qa` (`-a` wie „all“) heraus. Welche Dateien im (nicht installierten) `wwwoffle`-Paket drin ist, verrät

```
rpm -qpl wwwoffle-2.6b-2.i386.rpm
```

(`l` wie „list“), während

```
rpm -qi wwwoffle
```

„Informationen“ zum installierten `wwwoffle` gibt.
Zu welchem Paket eine Datei gehört, lässt sich dem Paketmanager mit

```
rpm -qf /etc/passwd
```

entlocken.

✎ Zu welchem Paket gehört das Kommando `ls`? Nenne fünf weitere Programme, die zu diesem Paket gehören! Lies die Paketbeschreibung.

**GNU-Make** Nicht zu jeder Software gibt es ein für die eigene Distribution geeignetes Binärpaket; viele Tools findet frau grundsätzlich nur im Sourcecode. Die entsprechenden, mit `gzip` oder `bzip2` komprimierten Quellarchive tragen meist Dateiendungen wie `tar.gz`, `.tgz` bzw. `.tar.bz2` oder `.tbz`. Sie gilt es zunächst auszupacken. Ein aktuelles `tar` vorausgesetzt, geht das mit `tar -xzf` bei mit `gzip` gepackten, mit `tar -xIf` bei mit `bzip2` gepackten `tar`-Archiven.
In aller Regel erhält frau ein neues Verzeichnis, in dem sich meist eine Datei namens `README` und/oder `INSTALL` befindet. Sie gibt Informationen darüber, wie frau zu einem Binärprogramm kommt. Einfachen Projekten liegt eine Datei `Makefile` oder `makefile` bei, die diese Informationen enthält, sodass ein einfaches `make` ausreicht, um den Kompiliervorgang einzuleiten. Oft ist es aber gerade in diesem Fall angebracht, einen Blick ins Makefile zu werfen, um ggf. den einen oder anderen Parameter anzupassen.
Sobald das Projekt komplexer wird, nach verschiedenen externen Bibliotheken und Hilfsprogrammen verlangt und gar noch portabel für verschiedene Plattformen sein soll, bietet ein festes Makefile zu viele Unwägbarkeiten. Dann (und das ist die Regel) liegt den Sourcen ein `configure`-Skript bei. Um es laufen zu lassen, wechselt frau ins Sourcenverzeichnis und ruft

```
./configure
```



# Samstag, 03. September 2005

## 7 Editoren

### 7.1 Vi

Ob E-Mails, Programme oder auch Dokumente wie dieses Skript – Textdateien werden mit *Editoren* erstellt bzw. verändert. In dieser Kategorie gibt es unendlich viele verschiedene Programme, aber nur eins, das auf (fast) allen Unixsystemen vorhanden ist: der `vi`, ein äußerst mächtiges Werkzeug zur Textbearbeitung. Seine Bedienung ist durchaus etwas gewöhnungsbedürftig, doch da er bei einigen (System-)Programmen als Standardeditor[20] dient, kann frau des öfteren in Situationen kommen, in denen sie wenigstens wissen muss, wie sie den Editor ungeschoren wieder verlässt.
Für dieses Programm gilt übrigens der Ausspruch „Liebe kommt mit der Zeit“[21]: Kaum eine, die ihren Vi heute nicht mehr aus der Hand legt, kam von Anfang an mit dem Programm klar ....
Weil der „Original“-Vi jedoch selbst Vi-Fans als recht sperrig erscheint, gibt es eine Menge *Klone*, die die Bedienung vereinfachen und zusätzliche Features wie Syntaxhighlighting eingebaut haben. Unter Linux wird es frau immer mit einem Vi-Klon zu tun haben – ob der `vim`, `elvis` oder `nvi` heißt, ist von Distribution zu Distribution verschieden und natürlich Geschmacksfrage. Auch wenn dieser auch unter seinem speziellen Namen ansprechbar ist, lässt sich einer der Klone immer mit dem Befehl

```
vi filename
```

aufrufen.
Gibt es die Datei *filename* bisher noch nicht, erscheint jetzt ein Bildschirm, in dem an jedem Zeilenanfang eine Tilde steht. Diese markiert eine Zeile und ist nicht wirklich in der Datei enthalten.
Dass Vi so gewöhnungsbedürftig ist, liegt an seiner Geschichte. Der Original-Vi war einer der ersten Editoren, mit denen sich eine ganze Datei auf einmal ändern ließ, statt zeilenorientiert vorzugehen wie bei den damals üblichen *Line Editors*. Von Zeileneditoren wie `ed` hat der Vi jedoch das Konzept der Unterscheidung zwischen Eintipp- und Bearbeitungsmodi geerbt:

**Kommandomodus:** Positionieren des Cursors, Texte suchen, Texte löschen usw.; aus dem Eingabemodus erreichbar durch Betätigen der (Esc)-Taste.

**Eingabemodus:** Texteingabe; im Kommandomodus eingegebene Texteingabekommandos wie (i) oder (a) schalten in ihn um.

**Ex-Modus:** Suchen und Ersetzen, Speichern und Beenden; eine Art interne Kommandozeile innerhalb des Kommandomodus, benannt nach dem Zeileneditor `ex`. Diese Kommandozeile erreicht frau aus dem Kommandomodus durch Eingabe von (:). In den Kommandomodus kehrt sie durch Abschicken des jeweiligen Befehls mit (↵) zurück.

[20] ..., der sich durch Umsetzen der Umgebungsvariablen `EDITOR` oder `VISUAL` umstellen lässt ...
[21] Silly: „Traumpaar“

Nach dem Aufruf befindet sich der `vi` in aller Regel zunächst im Kommandomodus. Zu den nun möglichen Befehlen zählen:

**Vi-Befehle**

| | |
|---|---|
| h, j, k, l | Umsetzen des Cursors auf die Position links neben, unter, über bzw. rechts neben der aktuellen (meist auch durch die Pfeiltasten möglich) |
| Strg+f | ganzer Bildschirm nach unten („forward"; meist auch durch Bild ↑ erreichbar) |
| Strg+b | ganzer Bildschirm nach oben („back"; meist auch durch Bild ↓ erreichbar) |
| ⇧+g | bewegt den Cursor an den Anfang der letzten Zeile der Datei |
| x | löscht das Zeichen unter dem Cursor |
| dw | löscht das Wort, auf dessen erstem Buchstaben der Cursor steht („delete word") |
| dd | löscht die Zeile, in der der Cursor steht |
| o | erzeugt eine neue Zeile unter der aktuellen und wechselt in den Eingabemodus |
| ⇧+o | erzeugt eine neue Zeile vor der aktuellen und wechselt in den Eingabemodus |
| ⇧+y | Kopieren der aktuellen Zeile in den Zwischenpuffer („yank") |
| p | Einfügen des Zwischenpufferinhalts hinter dem Cursor („print") |
| ⇧+yp | Duplizieren der aktuellen Zeile |
| u | Undo |
| /*Stichwort* | Vorwärtssuche nach *Stichwort* |
| ?*Stichwort* | Rückwärtssuche nach *Stichwort* |
| Strg+r | Rückgängigmachen eines Undo („redo") |
| :.s/*Suchbegriff*/*Ersatz*/g | Wechseln auf die interne Kommandozeile, anschließend Suchen und Ersetzen („substitute") in der aktuellen Zeile (ohne Nachfrage) |
| :%s/*Suchbegriff*/*Ersatz*/g | Suchen und Ersetzen in der gesamten Datei (ohne Nachfrage) |
| :.,$s/*Suchbegriff*/*Ersatz*/gi | Suchen und Ersetzen von der aktuellen Zeile bis zum Dateiende (mit Nachfrage) |
| :w | Speichern der Datei („write") |
| :q | Verlassen der Datei, wenn alles gespeichert ist oder nichts verändert wurde („quit") |
| :q! | Verlassen der Datei ohne Speicherung vorangegangener Änderungen |
| :wq | Speichern und Verlassen der Datei |

Wie schon in diesen Beispielen zu sehen, muten Vi-Befehle Nicht-Vi-Benutzerinnen unglaublich komplex an. So kann frau Befehlen im `ex`-Modus einen Bereich voranstellen, zum Beispiel

`2,6` – von Zeile 2 bis einschließlich Zeile 6,





Das Homeverzeichnis von `bille` tastet das Kommando nicht an, sodass ihre Daten nicht verloren gehen. Hier muss die Superuserin bei Bedarf selbst Hand anlegen.

✎ Wie würdest Du als `root` Billes Homeverzeichnis löschen?

Wie zum Anlegen neuer User gibt es natürlich auch Tools, die neue Gruppen erstellen (`groupadd`) oder eine Gruppe löschen (`groupdel`). Auf jeden Fall lohnt sich immer ein Blick in die passende Manpage.

## 8.2 Installieren von Software

Das Einspielen neuer Programme auf einem Rechner ist alles andere als eine triviale Angelegenheit: Da sollten die benötigten Bibliotheken in einer passenden Version installiert sein, möglicherweise wird weitere Software zum Funktionieren benötigt, und wenn frau das Programm doch wieder loswerden will, soll möglichst alles vom Rechner verschwinden, was es beim Installieren mitgebracht hat.
Um all diese Dinge kümmert sich bei den meisten Distributionen ein Paketmanager. Meist kümmert der sich um – für diese spezielle Distribution – vorkompilierte Binärpakete, es gibt aber mittlerweile eine Reihe Distributionen, die sich das *Ports-System* von (Free)BSD zum Vorbild nehmen, das dafür sorgt, dass Source-Pakete direkt auf dem jeweiligen Rechner konfiguriert, kompiliert und installiert bzw. entfernt werden.
Die weiteste Verbreitung hat der *Red Hat Packet Manager* `rpm` gefunden, der nicht nur bei Red Hat und Fedora, sondern auch von SuSE oder Mandrake eingesetzt wird. Technisch mit mehr Lorbeeren bedacht wird der Paketmanager der Debian-Distribution und ihrer Abkömmlinge, `dpkg`. Er brilliert vor allem durch sein Frontend `apt-get`, mit dem sich das System einfach übers Netz updaten lässt. Das Tool `alien` wandelt einfache `rpm`- und `deb`-Pakete bis zu einem gewissen Grad ineinander um.
Das Bauen der Binärpakete für einen Paketmanager ist eine Wissenschaft für sich und muss für jede Distribution (und jede Plattform) neu vorgenommen werden. Da die verschiedenen Versionen einer Distribution meist auf unterschiedlichen Bibliotheksversionen aufbauen, deren Schnittstellen oft nicht kompatibel sind, und auch die Distributoren selbst genügend einschneidende Veränderungen vornehmen, ist es oft nur möglich, ein Binärpaket einzuspielen, dass für die vorliegende Version einer Distribution gedacht ist.
Auch wenn mehrere Distributionen auf `rpm` setzen, heißt das daher nicht, dass Pakete distributionsübergreifend austauschbar wären. Während Red Hat/Fedora und Mandrake recht kompatibel zueinander sind, sodass sogar so manches Serverpaket der einen auf der anderen Distribution benutzt werden kann, ist bei SuSE höchstens bei Anwendungen an einen Austausch mit anderen Distributionen zu denken.
Im Folgenden widmen wir uns der Bedienung von `rpm`. Der Aufruf dieses Programms ist auch bei SuSE möglich, die zum Einspielen und Deinstallieren eigentlich Yast(2) propagiert. Letzterer wirkt in diesem Fall als `rpm`-Frontend[24].

**Red Hat Packet Manager** `rpm`-Pakete wie `wwwoffle-2.6b-2.i386.rpm` enthalten (mit Ausnahme von SuSE-Paketen) in ihrer Bezeichnung den Namen (`wwwoffle`[25]) und die Version (`2.6b`), das *Patchlevel* (`2`, eine Information darüber, wie oft es von Distributorenseite Änderungen am Paket gab) sowie die Ziel-Plattform (`i386`). Bei Quelltextpaketen fällt letztere natürlich weg.
Widmet sich frau einem bereits installierten Paket, muss sie den Namen (und darf die Version) angeben (`wwwoffle-2.6b` oder `wwwoffle`). Kümmert sie sich um ein noch nicht installiertes

[24] ... das sich leider nicht immer um Abhängigkeiten schert...
[25] Wer wissen will, was WWWoffle ist: http://www.gedanken.demon.co.uk/wwwoffle/

**Passwörter** Zwar werden die Passwörter mit ihrer Festlegung in `/etc/passwd` verschlüsselt abgelegt, doch da die Verschlüsselungsmethode bekannt ist, kann frau die Wörterbücher dieser Welt nehmen, auf dieselbe Art und Weise verschlüsseln und dann mit den Einträgen in der `passwd`-Datei vergleichen. Diese muss für alle User lesbar sein, damit sie ihr Passwort mit `passwd` ändern können.
Tools wie `crack` automatisieren diese Brut-Force-Methode (und dehnen sie auf gängige Modifikationen bei Wörterbuchpasswörtern aus). So kann jede, die einen Account auf dem Rechner hat, theoretisch recht unproblematisch nach unbedacht gewählten Passwörtern fahnden. Systemadministrator(inn)en, die größere Userpools zu verwalten haben, benutzen `crack` o. ä., um zu verhindern, dass durch derart schwach geschützte Accounts das gesamte System gefährdet wird, denn ein Rechner lässt sich viel einfacher von innen als von außen hacken.
Heutzutage ist es daher eher unverantwortlich, die Passwörter in der `/etc/passwd` abzulegen (auch wenn es sich nicht immer – z. B. bei der Verwendung von NIS – vermeiden lässt). Das Problem wird durch die Einführung von *Shadow-Passwörtern* aus der Welt geschafft.
Wenn in der zweiten `passwd`-Spalte ein `x` steht, wurde das Passwort in eine nur für `root` lesbare Datei namens `/etc/shadow` ausgelagert, die (wie `man 5 shadow` zeigt) auch weitere Möglichkeiten der Userverwaltung wie auslaufende Passwörter oder zeitgesteuertes Disablen eines Accouts erlaubt.

✎ Werden auf Deinem Rechner Shadow-Passwörter verwendet? Wenn ja, versuch einen Blick in die `/etc/shadow` zu werfen!

✎ Welche UserID hat `root`, welche hast Du? Welchen Gruppen bist Du zugeordnet?

**User und Gruppen modifizieren** Natürlich lassen sich die Einträge in `/etc/passwd`, `/etc/shadow`, `/etc/group` mit einem Text-Editor ändern, das Homeverzeichnis mit `mkdir` anlegen und mit `chown` an die neue Userin übergeben sowie mit `passwd` ein Anfangspasswort setzen.
Dennoch schätzen viele Sysadminen den gewissen Komfort, den distributionseigene Tools bei der Useradministration bieten. Mit dem im Kurs vermittelten Wissen sollte deren Bedienung in Eigenregie machbar sein. Wir konzentrieren uns daher an dieser Stelle auf Tools, die sich auf der Kommandozeile bedienen lassen und oft (wenn auch nicht immer) vorinstalliert sind.
Zum Einrichten neuer User gibt es (in der Regel in `/usr/sbin` installiert, einem Verzeichnis, das oft nicht im Default-Suchpfad steht) normalerweise das Kommando `useradd`, dem `root` auf der Kommandozeile die Parameter für die neue Userin mitgeben muss:

```
/usr/sbin/useradd -c "Sibylle Naegle" -d /home/bille -g 100 -p passwort
-s /bin/bash -u 2000 -m bille
```

Die Option `-m` bedeutet dabei, dass das Homeverzeichnis angelegt wird und in `/etc/skel`[23] stehende Konfigurationsdateien hineinkopiert werden. Lässt frau die eine oder andere Option weg, so werden Defaultwerte benutzt.

✎ Finde mit Hilfe der Manpage heraus, wofür die übrigen Optionen stehen!

Etwas komfortabler, da interaktiv, lassen sich User mit `adduser` anlegen.
Seltener gebraucht wird das `useradd`-ähnliche Programm `usermod` zum Modifizieren von Accountdaten. Zum Löschen eines Users kann `root userdel` benutzen:

```
/usr/sbin/userdel bille
```

[23] für „skeleton“





`.` – aktuelle Zeile,

`.-3,$` – drei Zeilen vor der aktuellen bis zum Dateiende,

`%` – gesamte Datei.

„Einfachen“ Befehlen im Kommandozeilenmodus kann sie wiederum die Anzahl der Wiederholungen mitgeben (`2Y` kopiert zwei Zeilen, `2dw` löscht zwei Wörter).
Das Suchen und Ersetzen funktioniert übrigens ähnlich wie in Perl.
Als Faustregel gilt: Nicht überwältigen lassen, denn es gibt Unmengen weiterer Befehle, die nicht einmal ständige Vi-Benutzerinnen alle kennen. Sie werden natürlich auch in der Manpage zu `vi` erklärt.

✎ Schreibe eine etwa fünf- bis zehnzeilige Datei, in der das Wort *Informatica* vorkommt. Kopiere eine Informatica-Zeile ans Ende der Datei. Lösche in einer Informatica-Zeile alle Wörter vor und hinter Informatica. Suche nach Informatica und ersetze alle Vorkommen durch *Informatica Feminale*. Speichere die Datei. Mach die Suchen-und-Ersetzen-Aktion rückgängig. Verlass die Datei, ohne sie zu speichern.

✎ Probiere die Kommandos `:!ls` und `:.!ls` aus und erkläre den kleinen Unterschied!

## 7.2 emacs

Unixerinnen lieben Glaubenskriege, und einer der ältesten und immer wieder gern mit Inbrunst geführten betrifft die Frage „Vi oder Emacs?“ Dank Menüführung fällt zumindest der Einstieg bei den verschiedenen Emacs-Varianten leichter (unter Linux findet frau meist den GNU- oder den XEmacs vor), und hinter der liebevoll-spöttischen Behauptung, Emacs sei kein Editor, sondern ein Betriebssystem, verbirgt sich die Wahrheit, dass Emacs dank verschiedener Erweiterungen weitaus mehr kann als das, was frau von einem Texteditor erwartet[22]. Wer den Emacs jedoch produktiv einsetzen möchte, kommt nicht darum herum, sich eine Menge nicht unbedingt leicht zu merkender Tastaturkürzel anzueignen.
Die Befehle, mit denen der Emacs gesteuert wird, beginnen immer mit (Strg) oder der auf der PC-Tastatur nicht vorhandenen Meta-Taste. Die Funktion der letzteren übernehmen in der Regel (Esc), (Alt) und/oder aber auch eine oder gar beide Windows-Tasten. Sowohl in der Literatur als auch im Emacs selbst findet frau dafür die Abkürzungen *M* für „Meta“ und *C* für „Control“, auf deutschen Keyboards also (Strg). Soll diese Taste gehalten bleiben, während die nächste gedrückt wird, so verbindet die Tasten ein Minus-Zeichen (-). Da diese Konvention so erdrückend verbreitet ist, folgen wir ihr der Gewöhnung halber auch in diesem Skript: *C-x C-s* bedeutet also, dass (Strg) so lange gehalten wird, bis frau auch (x) und darauf folgend (s) gedrückt hat.
Braucht der Emacs eine Zusatzinformation oder Bestätigung bzw. gibt er eine Rückmeldung, so erscheint der entsprechende Dialog am unteren Rand des Emacs-Fensters im sogenannten *Minibuffer*. Nach dem Befehl zum Speichern einer Datei erscheint darin beispielsweise die Rückmeldung:

```
Wrote /home/sibylle/informatica/linuxkurs.tex
```

Nach dem Befehl zum Öffnen einer Datei erscheint der aktuelle Pfad, an den ein Dateiname angehängt oder der auf den gewünschten Pfad/Dateinamen geändert wird. In diesem Moment gehen alle Tastatureingaben an den Dialog und nicht mehr in den eventuell bereits bearbeiteten Text, in Emacs-Parlance *Buffer* genannt. Natürlich muss niemand alle Dateien auswendig können:

[22] Welche es nicht glaubt, darf gerne *M-x doctor* ausprobieren ...

Mit (Tab) komplettiert Emacs bei eindeutigen Verhältnissen die Eingabe oder öffnet auf ein zweites (Tab) den sogenannten **Completions**-Buffer, in dem (nach Hineinwechseln mit *M-v*) eine Auswahl möglich ist.
Buffer mit solchen, von Sternchen umrahmten Namen sind Hilfs„ablagen“ des Programms. Jede geöffnete Datei wird durch einen Buffer mit ihrem Namen repräsentiert. Wichtig zu wissen ist dabei, dass ein Buffer einerseits in mehreren Unterfenstern (Ansichten) gleichzeitig bearbeitet werden, andererseits auch gerade in keinem Fenster sichtbar sein kann.

**Emacs-Tastenkombinationen**

| | |
|---|---|
| C-x C-c | Emacs beenden |
| C-x C-f | Datei öffnen |
| C-x C-s | Datei speichern |
| C-x C-w | Datei unter anderem Namen speichern |
| C-g | Befehl abbrechen |
| C-x u | Undo |
| C-x 2 | zweites Fenster unter dem aktuellen öffnen |
| C-x o | zwischen den Subfenstern wechseln |
| C-x 0 | aktuelles Subfenster schließen (Datei-Buffer bleibt erhalten) |
| C-x k | Datei schließen |
| C-s | nach Zeichenkette (inkrementell) vorwärts suchen |
| C-r | nach Zeichenkette (inkrementell) rückwärts suchen |
| C-e | zum Zeilenende |
| C-a | zum Zeilenanfang |
| M-→ oder M-← | zum nächsten Wort |
| M-> | ans Ende der Datei |
| M-< | an den Dateianfang |
| C-w | (zum Beispiel mit der Maus) markierten Textteil löschen und zwischenspeichern |
| C-y | zwischengespeicherten Textteil einfügen |
| M-% *vorher* (↵) *nachher* (↵) | Suchen und Ersetzen: auf die Nachfrage mit (y) für Ersetzen bzw. (n) für Nicht-Ersetzen antworten |
| M-x shell | Eine Shell öffnen |
| C-h t | Emacs-Tutorium starten |
| C-h i | Emacs-Info-System starten |

Tastenkombinationen sind übrigens nichts unabänderliches im Emacs: Die oben genannten sollten zwar überall funktionieren, doch ist frau niemals davor gefeit, dass der Distributor (oder die Systemadministratorin) das eine oder andere umkonfiguriert hat. Manche Befehle sind auch über weitere Belegungen erreichbar und alle Emacs-Kommandos grundsätzlich über die Eingabe von *M-x* und ihres Namens, doch das würde an dieser Stelle zu weit führen ...

✎ Lege eine Datei an, in der Du Dich den anderen Kursteilnehmerinnen vorstellst, und speichere die Datei in Deinem Homeverzeichnis!

✎ Starte das emacs-Tutorial, und probiere es einmal aus!





### 7.3 Weitere Editoren

Editoren gibt es wie Sand am Meer, sodass frau zumindest an selbstadministrierten Rechnern genau den benutzen kann, der ihr am besten zusagt. Häufig installiert sind:

1. `pico`, der (nicht-grafische) Editor, der zum Mailprogramm `pine` gehört
2. `joe`, ein beliebter nicht-grafischer Editor
3. `nedit`, ein Editor für X
4. die mit KDE mitgelieferten Editoren `kwrite` und `kate`
5. `gedit`, ein Editor aus dem GNOME-Projekt
6. `jove`, ein Emacs-ähnlicher, nicht-grafischer Editor

✎ Suche im Web nach weiteren Texteditoren und prüfe, welche dieser und der oben angegebenen Editoren auf Deinem Pool-Rechner installiert sind.

✎ Teilt die installierten Editoralternativen unter Euch auf und erstellt eine Kurzbeschreibung und Bedienungsanleitung für die anderen Kursteilnehmerinnen.

## 8 Crashkurs für Hobby-Sysadminen

Im Folgenden werden wir einige häufig anstehende Systemadministrationsaufgaben besprechen. Mangels `root`-Zugang können wir viele allerdings nicht in der Praxis üben.

### 8.1 Benutzerinnenverwaltung

Wenn neue Benutzerinnen auf einem Unixrechner arbeiten sollen, benötigen sie einen *Account*, also einen Usernamen, ein Passwort, eine Login-Shell und ein Homeverzeichnis. Diesen muss die Systemadministratorin anlegen. Sie muss aber auch dafür sorgen, dass unbenutzte Nutzerkonten entfernt werden, denn jeder nicht nötige Account ist ein Sicherheitsrisiko.
Die Angaben zu Username, Passwort, Homeverzeichnis, aber auch zu Login-Shell und Primärgruppe werden in der Datei `/etc/passwd` von Doppelpunkten getrennt in folgender Reihenfolge abgelegt:

```
root:x:0:0:root:/root:/bin/zsh
trish:x:1001:100:Patricia Jung:/home/trish:/bin/bash
Username:Passwort:UserID:GruppenID:Beschreibung:Homeverzeichnis:Login-She
ll
```

Welcher Gruppe die dort angegebene Primärgruppen-ID entspricht, steht in `/etc/group`. Wenn User Mitglieder in mehr als einer Gruppe sein sollen, ergänzt die Systemadministratorin dies in dieser Datei:

```
users::100:testuser
```

Außer `trish` ist also auch `testuser` Mitglied der Gruppe `users`.