LINUX Einstieg

Thomas Schanz
IAAT - Universität Tübingen

4. September 2019

Inhaltsverzeichnis

1 Vorwort

Hallo? Ja? Megadodo Publications, Verlag von 'Per Anhalter durch die Galaxis', dem absolut großartigsten Buch im ganzen Universum, kann ich was für sie tun?

Bitte?

Ja, Ich habe ihre Nachricht ausgerichtet, fürchte aber Mr. Zarniwoop ist zu beschäftigt um sie zu sehen, er macht gerade eine Kreuzfahrt durch die Galaxis.

Ja, er ist in seinem Büro, aber er macht gerade eine Kreuzfahrt durch die Galaxis.[1]

LINUX ist ein Universum auf Ihrem Schreibtisch.

Selbst nach 20 Jahre mit LINUX, kann LINUX immer noch überraschen und man entdeckt immer noch neues von Zeit zu Zeit. LINUX ist ein Universum auf Ihrem Schreibtisch. Wenn Sie verstehen wollen wie Computer wirklich funktionieren, dann sollten Sie Zeit mit LINUX verbringen. LINUX ist vollkommen offen. Sie können sich jedes Detail von LINUX anschauen und Sie können selbst an der Weiterentwicklung von LINUX mitarbeiten. LINUX stellt Ihnen dazu alle Werkzeuge zur Softwareentwicklung kostenlos zur Verfügung.

Aber auch wenn Sie Computer nur anwenden möchten und keine Software entwickeln, sollten Sie einen Blick auf LINUX riskieren. Abgesehen dass LINUX kein Geld kostet ist es eines der stabilsten und sichersten Betriebssysteme überhaupt. Es gibt zahllose kostenlose Programme für LINUX, und hinter LINUX steht keine Organisation oder Firma die Sie ausspioniert oder Ihre Daten sammelt! Sie werden sehen, dass sich der am Anfang etwas höhere Zeitaufwand für LINUX am Ende bezahlt macht.

Diese Anleitung ist für diejenigen gedacht, die mit LINUX zum ersten mal in Kontakt kommen, sei es als User oder sei es als Admin vielleicht der ersten eigenen Installation. Es sollte Ihnen klar sein, dass diese Anleitung nur an der Oberfläche kratzt.

Have a lot of fun!

[1] Douglas Adams - Per Anhalter durch die Galaxis

2 Einleitung

power On a new adventure

LINUX ist ein offenes, netzwerkfähiges, Multiuser-Vollmultitaskingbetriebssystem, das Multiprozessorbetrieb und Echtzeitausführung unterstützt. LINUX ist eine kostenlose Variante von UNIX, das 1969 von den BellLabs für den Betrieb auf Groß- und Supercomputern entwickelt wurde und das in seiner 50 jährigen Geschichte Beiträge von zahllosen IT-Firmen erhalten hat, unter anderen: SCO, IBM, SGI, NEXT, DEC, APPLE, Sun-Microsystems und Hewlett-Packard.

... aktuelle LINUXe sind in der Regel 64 Bit. Es gibt zwei Familien, die weite Verbreitung haben:

- BSD basierte (Berkeley) LINUXe
  (Debian/Ubuntu/Xandros/Knoppix)
  dpkg - Package Manager, apt- Advanced Package Tool, synaptic

- SystemV basierte LINUXe
  (Fedora/RedHat/Suse/Mandrake/Scientific-LINUX)
  rpm - Package Manager, yum - Yellowdog Updater Modified, yumex

daneben bestehen nach wie vor noch kommerzielle UNIXe, wie AIX (IBM), SOLARIS (ORACLE) und HP-UX (HP).

3 Betriebssysteme

Das Betriebssystem erfüllt folgende wichtige Aufgaben:

Das Betriebssystem des Computers ist die Schnittstelle zwischen dem Programm und der Hardware des Computers. Den Aufbau des Rechners kann man sich wie ein mehrstöckiges Haus vorstellen wie es in Abbildung 1 dargestellt ist.

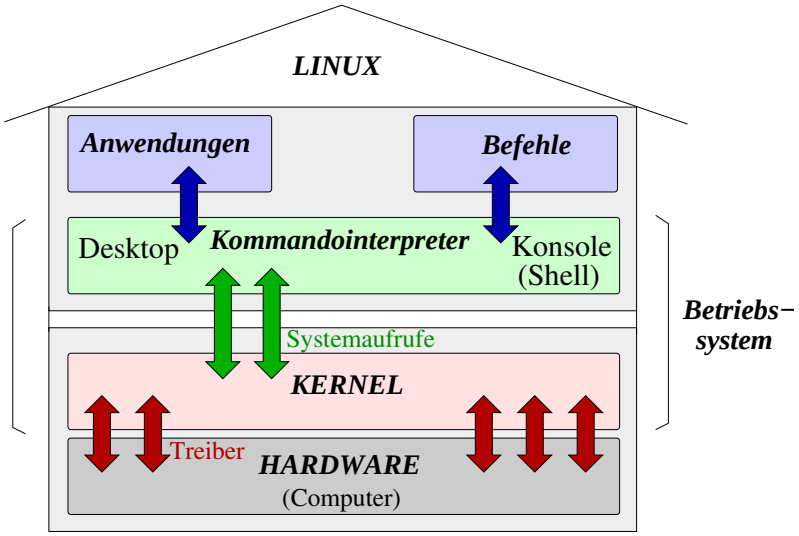

Abbildung 1: *Der Aufbau des LINUX-Betriebssystems entspricht einem mehrstöckigen Haus. Der KERNEL ist das Herz des Systems! Er steuert die Hardware über Treiber und koordiniert Systemaufrufe der Kommandointerfaces (Desktop und Konsole) und der verschiedenen Anwendungsprogramme.*

Im Keller ist die Hardware, Prozesseinheit, Speicher, Plattensysteme, Grafikkarte, usw. Darüber, im Erdgeschoss, befindet sich das Betriebssystem, es steuert die darunterliegende Hardware über so genannte Treiber. Von den oberen Stockwerken, wo sich die Programme befinden, empfängt das Betriebssystem Systemaufrufe, Programme können nur über das Betriebssystem auf die Hardware des Computers zugreifen! Die wesentlichen Hauptaufgaben des Betriebssystems sind:

1. es definiert eine eindeutige genormte Schnittstelle über die Programme mit der Hardware des Computers kommunizieren. Dies ist vor allem für Programmentwickler von herausragender Bedeutung
2. es koordiniert den Zugriff des Programms auf die Hardware des Computers
3. es koordiniert den Betrieb mehrerer gleichzeitig laufender Programme auf die Hardware des Computers
4. es trennt die Daten von verschiedenen Benutzern
5. es bietet eine Reihe von Werkzeugen zur Dateimanipulation

Im Prinzip könnten Computer auch ohne Betriebssysteme auskommen. Dann müsste das Programm aber die genaue Funktion der Hardware kennen, eine Aufgabe die bei modernen Computern von keinem Programmierer zu erfüllen ist. Außerdem kann in diesem Fall nur dieses einzige Programm gleichzeitig auf dem Computer laufen. Kleine Mikrocontrollersysteme laufen zum Beispiel auch heute noch ohne Betriebssysteme. Das Programm wird dort speziell für die jeweilige Hardware entwickelt.

- Moderne Betriebssysteme wie LINUX sind **'multiuserfähig'**. Dies bedeutet, dass mehrere Personen (User) einen Login auf der Maschine haben können. Diese Personen können den Computer nicht nur gemeinsam, sondern auch gleichzeitig verwenden! Das heißt es können mehrere User gleichzeitig auf ein und demselben System arbeiten ohne sich gegenseitig zu beeinflussen. Damit das möglich ist, muss das System netzwerkfähig sein.
- Ein **'netzwerkfähiges'** Betriebssystem wie LINUX erlaubt den 'Login' auch remote über ein Netzwerk. Benutzer können sich via Netzwerk anmelden und eine Session starten oder fortsetzen. Die Ausgabe erfolgt dann remote auf einem entfernten Terminal oder der Grafik eines entfernten, ans Netzwerk angeschlossenen Computers. Bei multiuserfähigen Betriebssystemen wird der Betrieb des Users, der an der Konsole sitzt, durch den Betrieb der über Netzwerk auf dem Rechner arbeitenden User nicht beeinträchtigt oder beeinflusst.
- Ein **'vollmultitasking'** Betriebssystem ist ein System auf dem mehrere Programme gleichzeitig auf ein und derselben Hardware laufen ohne sich gegenseitig zu beeinflussen. Jeder User kann also beliebig viele Programme gleichzeitig starten und ausführen. Die Programme können gleichzeitig die Hardware des Computers steuern, die Koordination übernimmt das Betriebssystem. Auch Multitasking ist eine Grundvoraussetzung für Multiuserfähigkeit!
- LINUX ist **'multiprozessorfähig'**. Es unterstützt die symmetrische Lastverteilung von Prozessen auf verschiedene Prozesseinheiten. Moderne Mehrcoresysteme profitieren davon auch ohne dass die Programme parallelisierbar sind, einfach indem bei Multitasking mehrere Programme symmetrisch auf die Cores verteilt wird.
- Manche LINUX-Systeme erlauben auch die **'Echtzeitausführung'** von Programmen. Diese Eigenschaft ist im Alltag meistens unbedeutend, Computer sind heute so schnell, dass die meisten Aufgaben scheinbar instantan erfolgen, obwohl der Scheduler des Betriebssystems Prozesse tatsächlich nacheinander und mit Pausen ausführt. Bei Echtzeitbetriebssystemen kann die Verarbeitung von Daten oder die Steuerung von Hardware zeitsynchron bis auf einzelne Taktzyklen festgelegt werden. Dies ist zum Beispiel von Bedeutung für Experimente in der Physik, bei denen Daten zeitlich lückenlos aufgezeichnet werden müssen.

Mit einem Computer werden
Fehler schneller als mit
jeder anderen Erfindung –
Mitch Ratcliffe

4 Login

Ist LINUX korrekt hochgefahren, meldet sich das System mit dem Login-Manager ähnlich wie in Abbildung 2.

Um sich anzumelden geben Sie ihren Login-Namen und das zugehörige Passwort ein und drücken Sie 'Enter' auf der Tastatur.

Das System meldet Sie mit einer Fensteroberfläche an, die Ihnen einen grafischen Desktop und ein Konsole-Interface zur Verfügung stellt. Dieser sogenannte Desktop-Manager erlaubt es nun per Mausklick Programme zu starten und mit diesen via Maus oder Tastatur zu kommunizieren. Ein File-Manager (Datei-Manager) ermöglicht das Erzeugen, Kopieren, Verschieben oder Löschen von Dateien. Ein Application-Manager erlaubt die Auswahl und das Starten der meisten Programme. Dies beschränkt sich allerdings auf Software die auf dem Grafiksystem läuft und eine Fensterausgabe erzeugt. Programme auf der Kommandozeile (Konsole) sind oft nicht sichtbar auf dem Application-Manager weil sie nur eine Kommandozeilenausgabe erzeugen.

Die Sitzung kann via Menüpunkt im Desktop-Manager beendet werden. Das System kehrt dann zum Login-Screen zurück. Nach dem Logout bleiben alle Benutzerdaten intakt und so erhalten wie Sie sie verlassen haben.

LINUX-Systeme sind Multiuserfähig! Per Default sind alle Benutzerdaten eines Login von allen anderen Benutzerdaten anderer Logins getrennt. Kein User kann die Dateien anderer User sehen oder aufrufen, außer Sie lassen dies explizit zu (siehe Dateirechte) oder es gehört zur Policy ihrer Arbeitsgruppe und wurde vom Admin anders eingerichtet.

Kein User kann System-Dateien, die zum LINUX-Betriebssystem gehören, verändern oder löschen. Sie können das System als User durch Unachtsamkeit nicht beschädigen, lediglich ihre eigenen Daten in Ihrem eigenen User-Account! Wenn Sie Dateien ausserhalb ihres User-HOME-Verzeichnisses auf dem System lesen, schreiben oder ausführen können, dann ist das so gewollt (gewöhnlich nicht der Fall)! Diese rigorose Trennung von User- zu User-Dateien und User- zu System-Dateien macht LINUX relativ unanfällig gegen Schadsoftware und Viren. Sie können ein LINUX-System als User nicht (naja sagen wir fast nicht) infizieren. Falls Sie das LINUX-System selbst betreiben und selbst Admin-Rechte auf dem System haben, dann sollten sie von diesen Admin-Rechten nur gebrauch machen, wenn es auch erforderlich ist! Machen Sie es sich nicht zur Gewohnheit sich auf dem Rechner als Admin anzumelden und alltägliche Aufgaben als Admin auszuführen. Es ist besser Sie melden sich als User an und geben sich Admin-Rechte in einer Konsole (Befehl ’su -’) wenn, und solange die Aufgabe dies erfordert.

```
Hinter jedem großen Computer
steckt ein kleiner knochiger
           Mann
```

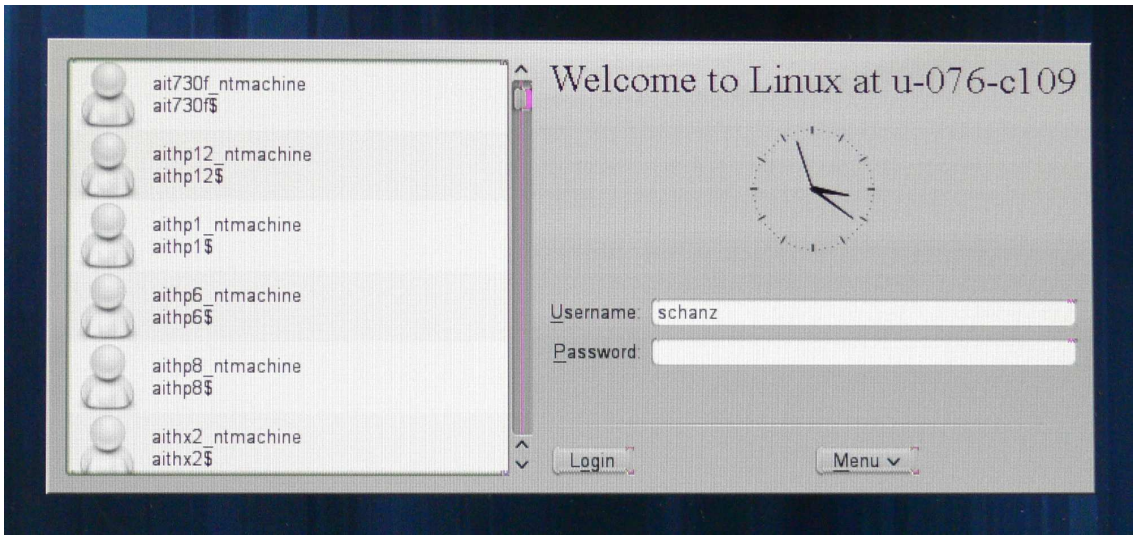

Abbildung 2: *Wie der Login-Screen genau aussieht hängt vom Login-Manager ab. Es gibt bei LINUX natürlich mehrere verschiedene Login-Manager. Allen gemeinsam ist, dass man sich per User-Name und User-Passwort anmelden kann. Viele Login-Manager bieten auch zusätzlich die Funktion, das System herunterzufahren (F11). Die Funktion soll es Usern ermöglichen die Maschine ordnungsgemäß herunterzufahren auch wenn der Login fehlschlägt. Sie sollten das natürlich nur ausführen wenn Sie sicher sind, dass niemand mehr auf der Maschine arbeitet, auch remote nicht.*

Seien Sie sich bewusst, dass Sie das LINUX-Betriebssystem (nicht die Hardware) zerstören können, wenn Sie als Admin angemeldet sind. Wenn Sie z.B. versehentlich den KERNEL löschen, startet ihr System nie wieder :-(.

LINUX-Systeme, die Sie nicht selbst verwalten, dürfen nicht heruntergefahren oder ausgeschaltet werden! Es können sich dutzende andere Benutzer auf dem System befinden die darauf arbeiten und wichtige Daten prozessieren. Das Herunterfahren des Systems führt dazu, dass diese anderen Benutzer ausgeloggt und ihre Programme und Prozesse zwangsweise vor ihrem normalen Ablauf beendet werden.

Schlimmer noch als das unberechtigte Herunterfahren des Systems ist das direkte Ausschalten des Computers! In diesem Fall hat das Betriebssystem keine Möglichkeit mehr Daten die sich im Hauptspeicher befinden auf Massenspeicher heraus zuschreiben. Ein echter Datenverlust ist die Folge, beschädigte Dateien und im schlimmsten Fall ein defektes Betriebssystem! Ein Defekt von Hardware kann aber selbst in diesem Fall nicht auftreten, auch wenn dies gelegentlich behauptet wird :-) .

Also Finger weg! Sie sollten nur LINUX-Rechner die Ihnen selbst gehören und die Sie selbst verwalten herunterfahren oder ausschalten, wenn Sie wissen was sie tun!

```
Am Ursprung von jedem Fehler,
  der dem Computer zur Last
   gelegt wird, findet man
 mindestens zwei menschliche
  Fehler, einschließlich des
  Fehlers die Schuld auf den
     Computer zu schieben
```

5 Benutzerinterface

Damit der User eigene Programme bzw. Prozesse starten kann benötigt er ein so genanntes Benutzerinterface. Dieses ist entweder eine grafische Benutzeroberfläche, die mit der Maus bedient wird, oder eine Terminal-Emulation mit einer Shell auf der Kommandos per Tastatur eingegeben werden können. Die beiden Möglichkeiten Prozesse zu starten sind also kurz gesagt der **'Desktop'** und die **'Konsole'** und beide in Abbildung 3 dargestellt.

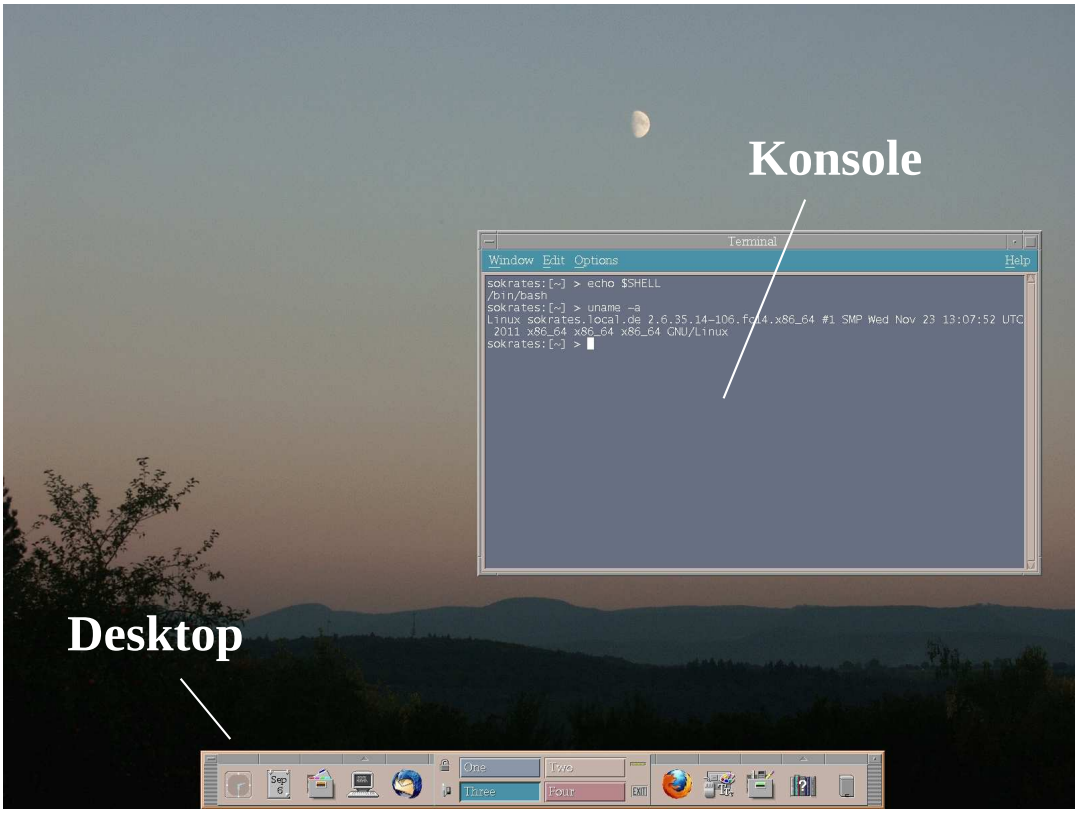

Abbildung 3: *Die beiden Benutzerinterfaces von LINUX: Desktop und Konsole. Im Prinzip kann man LINUX so konfigurieren das es ausschließlich mit einer Konsole startet -ohne Desktop-, früher war das der Standard. Heute wird immer zuerst der Desktop gestartet. Der Desktop bietet die Möglichkeit eine oder sogar mehrere Konsolen gleichzeitig zu starten. Der Screenshot zeigt ein CDE-Desktop mit einem Bedienpult aus dem die Konsole und andere Programme gestartet werden können.*

Beide diese Benutzerinterfaces besitzen ihre Vor- und Nachteile. Der Desktop ist (idealerweise) intuitiv und leicht zu bedienen, Prozesse werden hier durch Mausklick gestartet. Der Desktop erleichtert es vor allem Anfängern auf einfache weise mit dem Computer zu arbeiten. Für den täglichen Gebrauch, z.B. zum eMail lesen oder surfen im Internet ist die Bedienung des Computers über das Desktop-Benutzerinterface ideal und der Konsole vorzuziehen.
Die Konsole hat indessen Vorteile wenn es um die automatisierte Steuerung von Prozessen geht. Das Benutzerinterface der Konsole verwendet statt der Maus viele Dutzend Befehle zum starten und zur Verwaltung von Prozessen, Kommandos die per Tastatur eingegeben werden müssen. Diese Kommandos sind im Grunde auch nur kleine Programme die auf dem System ausgehführt werden. Ihre große Anzahl und Vielfalt – viele besitzen zahlreiche Optionen – machen zum Teil den Ruf von LINUX als schwer erlernbares Betriebssystem aus. Es ist aber gerade diese Vielseitigkeit und Erweiterbarkeit des Konsole-Kommandointerpreters, die LINUX auch unvergleichlich leistungsfähig und seinen großen Reiz ausmachen. Die meisten Profis bevorzugen die Konsole. Abbildung 6 zeigt das Fenster einer typischen Konsole durch die der Computer ausschließlich über Befehle bedient werden kann.

Ein Beispiel: Angenommen es besteht die Aufgabe in einem Dateisystems mit 600 Unterverzeichnissen alle Dateien mit der Endung '.tex' nach dem String „AIT“ zu durchsuchen und diesen String durch den String „IAAT“ zu ersetzen. Um diese Aufgabe vom Desktop aus mit der Maus zu erfüllen müssten Sie also alle 600 Unterverzeichnisse aufrufen und jede Datei mit der Endung *.tex mit einem Editor öffnen um dort durch die „Search & Replace“ Funktion den String „AIT“ mit „IAAT“ zu ersetzen. Anstatt das Wochenende mit dieser Tätigkeit zu verbringen kann die gleiche Aufgabe mit der Konsole in wenigen Sekunden erreicht werden, z.B. mit der Kommandozeile:

```
[~]> for i in 'find . | grep .tex' do; sed -i -e "s/AIT/IAAT/g" $i; done'.
```

Hierin steckt die Leistungsfähigkeit des Kommandointerpreters der Konsole, er bietet ein programmierbares Interface!

5.1 Der Desktop

Es gibt eine ganze Reihe von verschiedenen Desktops für LINUX. Einige ahmen die Oberfläche von MICROSOFT WINDOWS nach und bieten die Möglichkeit Programme über ein Menüsystem zu starten. Andere verwenden eine Art Bedienpult über das auf alle täglich wichtigen Funktionen des Computers direkt zugegriffen werden kann. Allen Desktops gemeinsam ist, dass sie einen Datei-Manager besitzen, über den auf alle Dateien und Verzeichnisse des Dateisystems via Maus zugegriffen werden kann. Programme werden entweder über ein Menüsystem gestartet, oder der Desktop bietet einen speziellen Application-Manager der ebenfalls mit der Maus bedient wird.

Es gibt eine Menge verschiedener Desktops für LINUX. Die am meisten verbreitetsten sind:

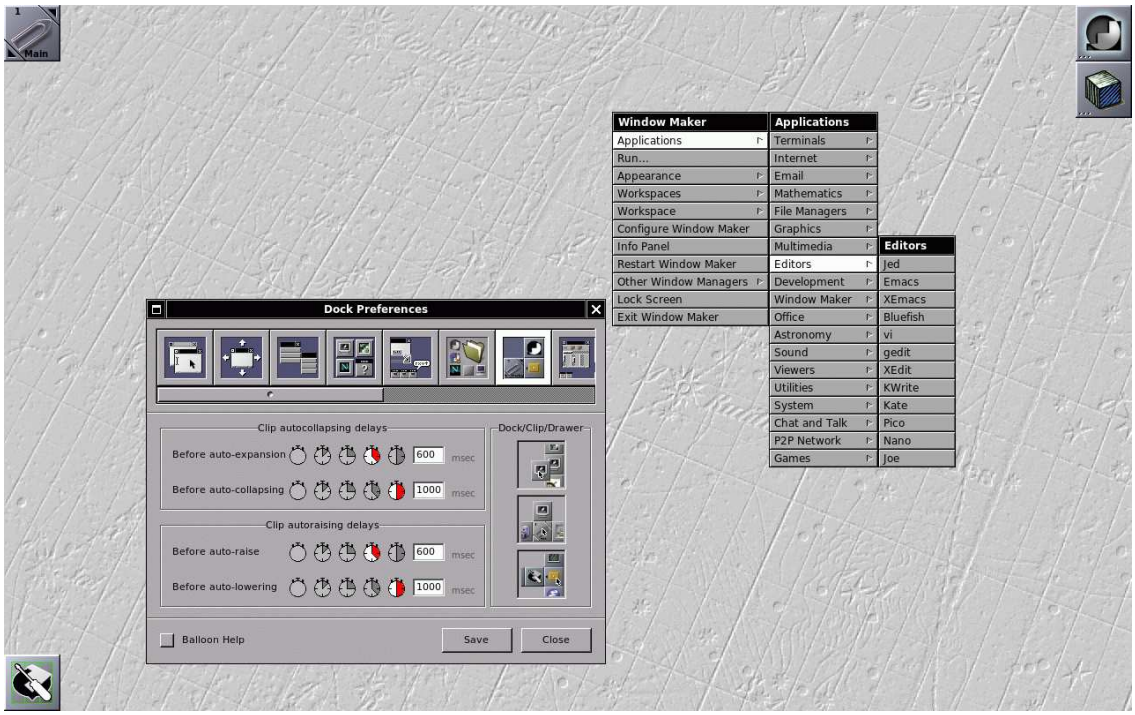

Abbildung 4: *Screenshot des 'WINDOWMAKER'-Desktop mit Konfiguration und Menüsystem.*

- KDE - Desktop:
  sehr leistungsstark und sehr umfangreich ausgestatteter Window-Manager, schwerfällig, benötigt viele Ressourcen
- GNOME - Desktop:
  Der Standard Window-Manager von Fedora-LINUX, sehr leistungsstark und sehr umfangreich ausgestattet, schwerfällig, benötigt viele Ressourcen
- XFCE - Desktop:
  leistungsstarker, gut ausgestatteter Window-Manager, Kompakt und schnell
- LXDE - Desktop:
  gut ausgestatteter Window-Manager, sehr kompakte und schnell
- UNITY - Desktop:
  der Ubuntu-LINUX Standard Window-Manager
- MATE - Desktop:
  kompakter und schneller Window-Manager
- CDE - Desktop:
  kompakter und schneller Window-Manager, Industriestandard auf UNIX Systemen über ein Jahrzehnt
- FVWM - Desktop:
  sehr einfacher Window-Manager, Sehr schnell und sehr kompakt, aus den Anfangsjahren von LINUX
- WINDOWMAKER - Desktop:
  sehr kompakter und schneller Window-Manager

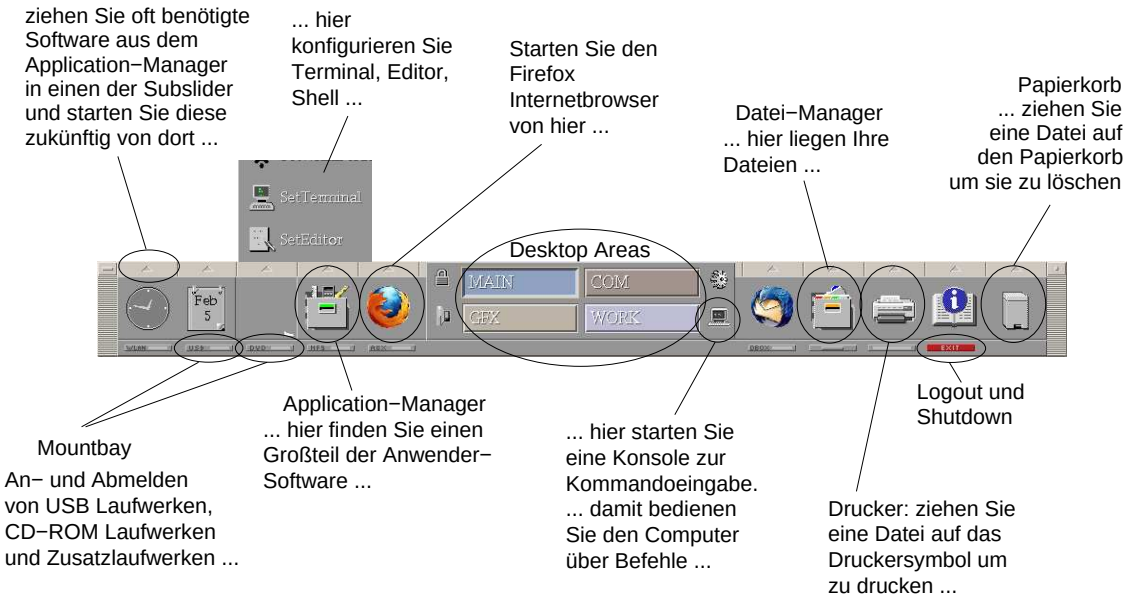

Abbildung 5: *Das Bedienpult von CDE bietet direkten Zugriff auf alle täglich wichtigen Funktionen des Computers.*

Die verschiedenen Erscheinungsformen von verschiedenen LINUX-Distributionen beruht hauptsächlich auf der Verwendung von unterschiedlichen Desktop-Managern. Viele Distributionen haben starke Vorlieben für spezielle Desktops. Unterhalb der Desktops sind sich die meisten LINUX-Distributionen hingegen sehr ähnlich, es ist die Wahl des Desktops, die LINUX so unterschiedlich aussehen läßt. Viele Desktops verwenden eine unterschiedliche Fensterdekorierung und unterschiedliche Fensterfunktionen (Widgets).

Bei den meisten LINUX-Distributionen lassen sich andere Desktops nachinstallieren. Der Benutzer kann dann den von ihm gewünschten Desktop im Login-Manager vor dem Login einstellen.

5.2 Die Konsole

Das andere Benutzerinterface ist die Konsole. Auf der Konsole-Ebene sind die meisten LINUX-Distributionen so gut wie identisch! Die Konsole wird lediglich von der Auswahl der Terminal-Emulation und der Shell beeinflußt, siehe hierzu auch Kapitel 15 und 14. Die Konsole startet ein Fenster auf dem Desktop, das ein programmierbares Kommando-Interface zum Computer bereit stellt, siehe hierzu auch Abbildung 6.

```
Entwerfe ein System das
selbst ein Idiot bedienen
kann und nur ein Idiot wird
es bedienen wollen
```

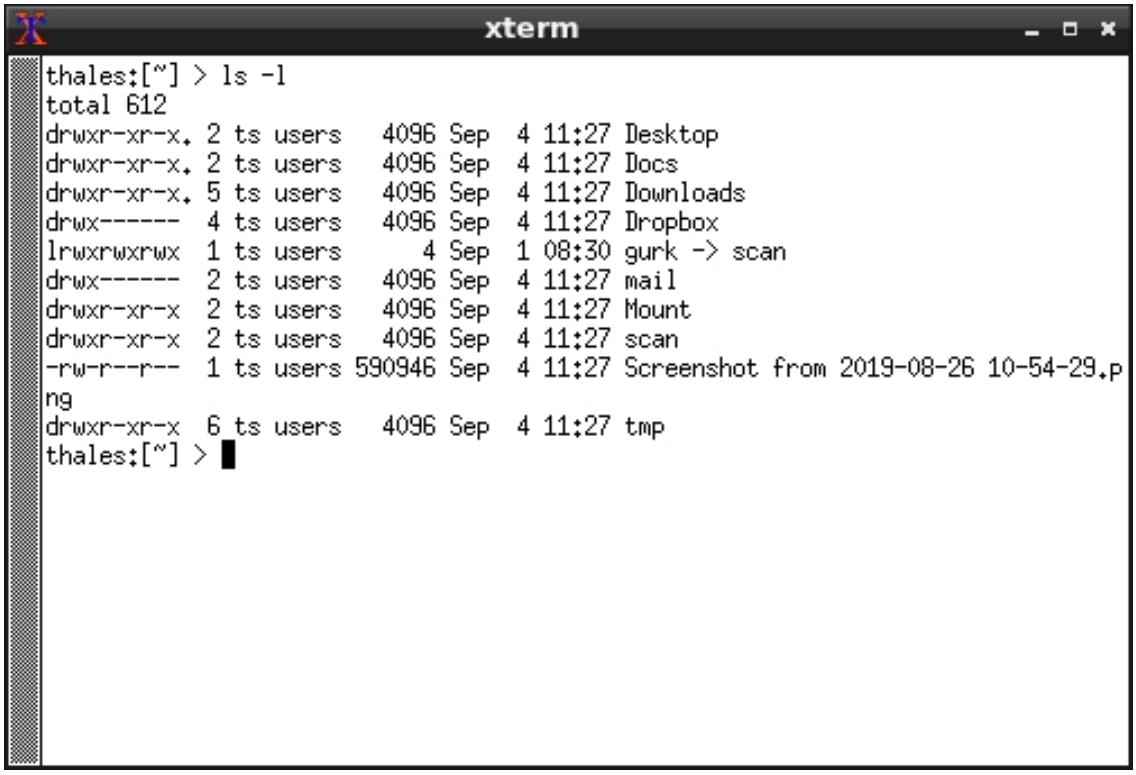

Abbildung 6: *Screenshot einer typische Konsole. Die Konsole zeigt hier die Ausgabe des 'ls -l' Befehls auf dem System 'thales', der im HOME-Verzeichnis von User 'ts' ausgeführt wurde.*

6 System-Shutdown

Wie wird ein LINUX-System ordnungsgemäß heruntergefahren?

Da LINUX ein Multiuser-System ist, kann nur der Systemverwalter einen Shutdown des Systems durchführen, normale User haben normalerweise nicht die Berechtigung!

Man kann LINUX-Systeme so aufsetzen, dass ein Shutdown des Systems über den Desktop-Manager der aktuellen Session auch durch normale User möglich ist. Bei Systemen die Sie Zuhause für sich alleine betreiben ist so eine Installation sehr sinnvoll.

Auf öffentlichen Systemen kann ein Shutdown nur mit Administrator-Rechten oder Super-User-Rechten durchgeführt werden. Hierzu rufen Sie eine Konsole bzw. ein Terminal auf und geben folgende Zeile ein:

```
[~]> sudo shutdown -h 0
```

(es wird das Passwd des Users verlangt)

Das '-h' steht für 'halt', die Zahl (hier '0') gibt die Anzahl von Minuten an, die Benutzer Zeit bekommen um ihre Arbeit zu beenden und alle Daten zu speichern bevor das System herunter fährt. Dazu werden im Minutentakt Shutdown-Warnungen an alle Konsolen aller

User auf dem System ausgegeben. Als System-Admin sollten Sie also auf keinen Fall hier ’0’ eingeben, wenn noch Benutzer auf dem System sind, ’30’ oder ’60’ sind meistens ausreichend. Bei der Eingabe von ’0’ fährt das System augenblicklich herunter. Wenn Sie alleine auf dem System arbeiten ist das prima.

Mit dem Befehl 'su -' können Sie auch dauerhaft SuperUser Privilegien erwerben (es wird das SuperUser Passwd verlangt). Gelingt das Anmelden als Superuser, so können Sie den shutdown-Befehl auch direkt eingeben:

```
[~]> shutdown -h 0
```

oder

```
[~]> reboot -h
```

ACHTUNG!
warten Sie bis die Ausgabe:

```
Halted, you may now cycle power
```

erscheint. Erst dann den Rechner ausschalten!

Moderne LINUX-Rechner fahren nach der Eingabe von 'shutdown' selbstständig herunter und schalten sich vollständig aus. Bei manchen Systemen fährt das LINUX nur bis zur Nachricht 'Halted, you may now cycle power' herunter. Das Ausschalten des Rechners muss dort manuell erfolgen.

ACHTUNG!
Das System immer sauber herunterfahren, niemals den Rechner einfach ausschalten! Die Zerstörung des Betriebssystems und/oder Datenverlust kann die Folge sein!

Brian Kernighan hat an einem Auto mitentworfen. Anders als die meisten Autos hat es weder Tacho, Tankanzeige, noch eine der anderen blinkenden Anzeigen die den Fahrer nerven. Wenn ein Fehler passiert, erscheint nur ein großes ’?’ in der Mitte des Armaturenbretts! ”Der erfahrene Autofahrer”, sagt er, ”weis für gewöhnlich dann genau was verkehrt war.”

7 Prozesse

Das Herz des LINUX-Betriebssystems ist der KERNEL. Der KERNEL ist ein einzelnes Programm (z.B. vmlinuz-4.8.13-100.fc23.x86_64 – liegt unter /boot des Dateisystems), das beim Systemstart aufgerufen und gestartet wird. Der KERNEL über- nimmt die gesamte Steuerung der Hardware und insbesondere das Prozess-Scheduling mit dem der Betrieb von Programmen auf der Computerhardware gesteuert wird.

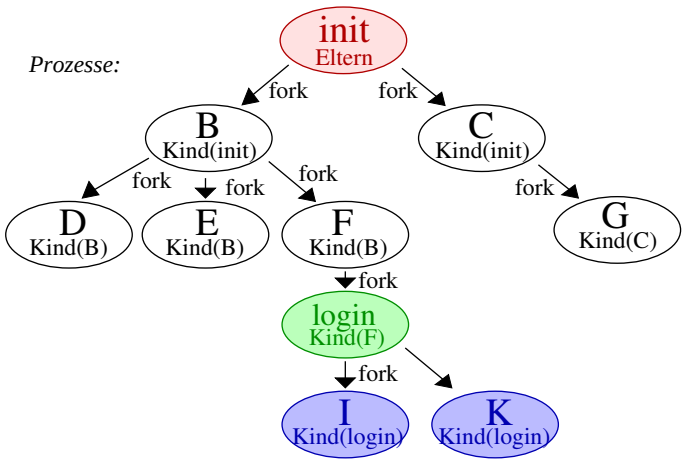

Abbildung 7: *Beim Booten startet mit dem KERNEL der erste Prozess des Systems mit der Prozess-ID Nummer '1', der 'init-Prozess'. Alle weiteren Prozesse entstehen aus 'init' durch Vergabelung (fork), 'init' ist der Eltern- Prozess des gesamten Systems. Alle anderen Prozesse (fast alle) sind Kind-Prozesse von 'init'. Mit dem Konsole-Befehl 'ps -fe' kann man sich die Liste (siehe unten auszugsweise) aller laufenden Prozesse anzeigen lassen, PPID ist darin die Prozess-ID Nummer des jeweiligen Eltern (Parent)-Prozesses. Prozess-ID Nummer '3730' ist in diesem Beispiel der Login-Prozess von User 'schanz'.*

Beim Booten des LINUX-Systems wird zuerst der KERNEL gestartet. Dieser startet zunächst den 'init'-Prozess. Was auf dem System eigentlich später läuft sind keine Programme, sondern Prozesse, Programme starten Prozesse! Der erste Prozess der mit dem KERNEL gestartet wird ist der 'init'-Prozess mit der Prozess-ID Nummer '1', vergleiche hierzu Abbildung 7.

Von diesem Prozess aus stammen alle späteren Prozesse ab, man spricht von Eltern-Prozess und Kind-Prozessen. Alle Kind-Prozesse erben die Aufrufparameter der Eltern-Prozesse. Wird der Eltern-Prozess beendet, sterben auch alle Kind-Prozesse (anders als im richtigen Leben). Der 'init'-Prozess ist also der Eltern-Prozess des gesamten Systems. Beim Booten vergabelt sich der 'init'-Prozess zu einer Vielzahl von Kind-Prozessen, die das LINUX-System zum Betrieb benötigt.

Ist das System vollständig gestartet, können sich Benutzer anmelden. Bei diesem Vorgang wird der Prozess des Loginmanagers weiter vergabelt, das Kind ist der User-Login-Prozess, der Eltern-Prozess aller Prozesse die der User später selbst starten wird. Wird

der User-Login-Prozess beendet, sterben alle laufenden User-Prozesse und der User wird automatisch ausgeloggt.

> Wie in gewissen Kulturen ist es erst dann möglich einen Prozess zu töten wenn sein wahrer Name bekannt ist. – Ken Thompson and Dennis M. Ritchie

Wird in der Konsole ein Programm, also ein Prozess gestartet, so vergabelt der Eltern-Prozess der Konsole und erzeugt einen Kind-Prozess. Der Eltern-Prozess wartet dabei solange bis der Kind-Prozess abgelaufen ist und setzt dann seine Arbeit fort. Der Kind-Prozess wird in diesem Fall als Vordergrundprozess bezeichnet. Dies ist der Normalfall.

Man kann Prozesse jedoch auch als Hintergrundprozess in der Konsole starten. Hierzu wird beim Aufruf einfach ein '&' hinter den Programm- oder Befehlsnamen gesetzt. Der Eltern-Prozess erzeugt dann einen Kind-Prozess, wartet aber nicht auf das Ergebnis des Kind-Prozesses und setzt seine Arbeit unabhängig fort. Der Aufruf:

```
[~]> firefox &
```

startet den ’Firefox-Webbrowser als Hintergrundprozess, die Konsole steht danach zu weiteren Eingaben zur Verfügung. Der Kind-Prozess (firefox) läuft in diesem Fall also im Hintergrund des Eltern-Prozesses (der Konsole) ab, deshalb die Bezeichnung ’Hintergrundprozess’.

Es ist möglich einen Prozess nachträglich in den Hintergrund zu verlegen. Hierzu gibt man in der Konsole, aus der heraus der Kind-Prozess gestartet wurde **'ctrl z'** ein, **'ctrl z'** suspendiert den Kind-Prozess. Danach kann man durch die Eingabe von **'fg'** (foreground) in der Konsole den Kind-Prozess wieder in den Vordergrund versetzten, also die Suspendierung aufheben oder durch Eingabe von **'bg'** (background) den Kind-Prozess als Hintergrundprozess abkoppeln, ganz so als hätte man ihn von vornherein mit '&' gestartet.

Die Eingabe von **'ctrl c'** in der Konsole beendet den zuletzt gestarteten Kind-Vordergrundprozess vollständig.

Mit dem Kommando **'ps'** bzw. **'ps -fe'** kann man sich die laufenden Prozesse anzeigen lassen. Auch das Programm **'top'** gibt eine Übersicht der laufenden Prozesse mit der höchsten CPU-Auslastung.

```
UID        PID  PPID  C STIME TTY      TIME CMD
root         1     0  0 08:54 ?    00:00:01 init [2]
:
root      1225     1  0 08:54 ?    00:00:00 udevd --daemon
daemon    2577     1  0 08:54 ?    00:00:00 /sbin/portmap
statd     2589     1  0 08:54 ?    00:00:00 /sbin/rpc.statd
root      2924     1  0 08:54 ?    00:00:00 /usr/sbin/rsyslogd -c3
root      2935     1  0 08:55 ?    00:00:00 /usr/sbin/acpid
root      3669  3662  0 08:55 ?    00:00:00 /usr/sbin/gdm
```

```
:
schanz    3730     1  0 08:55 ?    00:00:00 /usr/bin/gnome-keyring-daemon -d --login
schanz    3731  3669  0 08:55 ?    00:00:00 /bin/sh /usr/bin/startkde
:
```

Über das Kommando **'kill'** können Prozesse gewaltsam beendet werden (siehe später).

8 Dateisystem Aufbau

Alle LINUX-Systeme benutzen einen hierarchischen Dateisystemaufbau, der die hier aufgeführte Struktur hat und in Abbildung 8 dargestellt ist. Das System ist über diesen Dateisystemaufbau organisiert. Es ist sinnvoll eine Vorstellung zu haben, wo welche Dateien liegen. Das Dateisystem enthält alle Dateien die auf dem LINUX-System existieren, es gibt keine Dateien die zum System gehören, sich aber nicht innerhalb des Dateisystems befinden. Das gilt auch für montierte Medien! Sobald ein Medium (z.B. ein USB-Stick) montiert ist, gehört es zum Dateisystem.

Alle Dateien des Dateisystems sind sowohl auf der Konsole wie auch für den Datei-Manager des Desktop sichtbar und erreichbar, dies ist in Abbildung 9 dargestellt. Es gibt beim Desktop-Datei-Manager jedoch die Möglichkeit Dateien, die mit einem Punkt '.' beginnen, auszublenden (das ist der Default). Sie können also jede Datei sowohl über die Konsole so wie auch über den Desktop aufrufen. Der Desktop-Datei-Manager erkennt Dateien anhand der Dateiendung und startet die zugehörige Anwendung wenn Sie auf das Piktogramm der Datei klicken. Das Kopieren und verschieben geschieht auf dem Desktop einfach in dem das Piktogramm der Datei mit der Maus gegriffen und bewegt wird. Zum Löschen steht ein Mülleimersymbol zur Verfügung auf das das Piktogramm der Datei einfach geschoben wird, 'drag & drop' heißt die Technologie dahinter. Dateien die Sie auf den Mülleimer schieben werden nicht direkt gelöscht, sondern nur in ein 'Mülleimerverzeichnis' verschoben, von wo Sie sie dann später gemeinsam löschen oder bei Bedarf wieder herstellen können. Erst wenn Sie den Mülleimer leeren werden die Dateien darin unwiderruflich gelöscht. Der Desktop soll hier jedoch nicht weiter besprochen werden, wenden wir uns also lieber wieder der Konsole zu ....

Alle Verzeichnisse von LINUX wurzeln im so genannten 'root'-Verzeichnis das durch das Symbol '/' gekennzeichnet ist. Pfadangaben die mit '/' beginnen werden als 'absolut' bezeichnet und sind immer eindeutig. Das Kommando 'cd /home/schanz/tmp' zum Beispiel wechselt den Kommandoprompt der Konsole in das tmp-Verzeichnis des Users 'schanz'. Für User 'schanz' hätte vermutlich auch die Eingabe 'cd tmp' ausgereicht. eine Pfadeingabe die 'relativ' ist und nur funktioniert wenn sich User 'schanz' im von 'tmp' aus gesehen richtigen Verzeichnis befindet (hier dem User-HOME-Verzeichnis '/home/schanz/'). Absolute Pfadangaben sind immer eindeutig, relative Pfadangaben erfordern, dass Sie sich im klaren sind von wo aus Sie navigieren. Verwenden Sie den Befehl 'pwd' um das festzustellen!

Geben Sie hierzu zum Beispiel folgendes in die Kommandozeile ein:

```
[~]> pwd
```

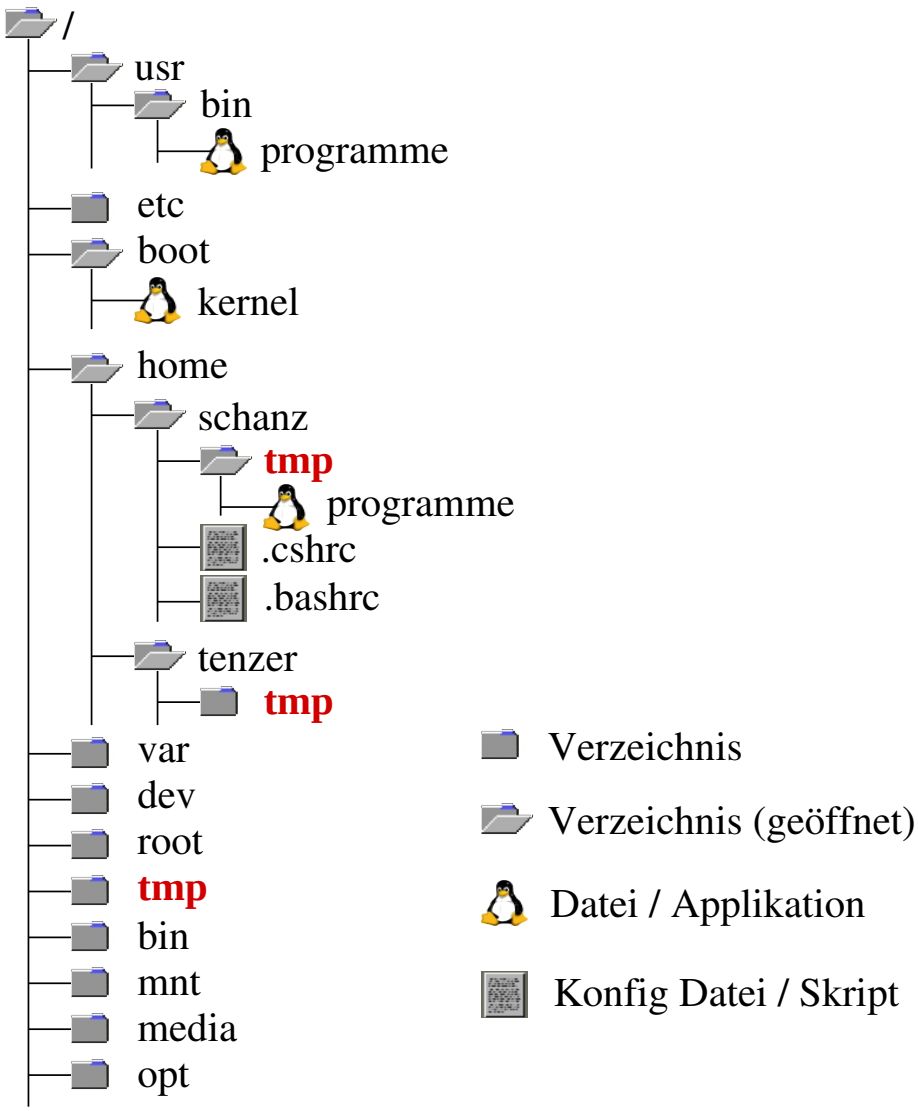

Abbildung 8: *LINUX hat einen hierarchischen Dateisystemaufbau. Das oberste Verzeichnis '/' heißt das 'root'-Verzeichnis des Systems, alle anderen Verzeichnisse verzweigen von hier. Das Dateisystem enthält alle Unterverzeichnisse und alle Dateien die auf dem LINUX-System existieren können, inklusive allen User-Dateien. Angaben für Unterverzeichnisse sind 'relativ', solange nicht der gesamte Pfad vom 'root'-Verzeichnis aus mit angegeben wird. Man beachte das z.B. unterschiedliche 'tmp'-Verzeichnisse im 'root'-Verzeichnis wie auch in den User-HOME-Verzeichnissen existieren. Um in das Verzeichnis '/home/schanz/tmp' zu wechseln müssen Sie entweder einen 'absoluten' Pfad angeben (z.B. 'cd /home/schanz/tmp/') oder Sie müssen wissen wo Sie sich 'relativ' zu '/home/schanz/tmp' im Dateisystem befinden. Angenommen Sie sind in '/home/tenzer/tmp', so ist der 'relative' Pfad nach '/home/schanz/tmp' z.B. gegeben als 'cd ../../schanz/tmp'.*

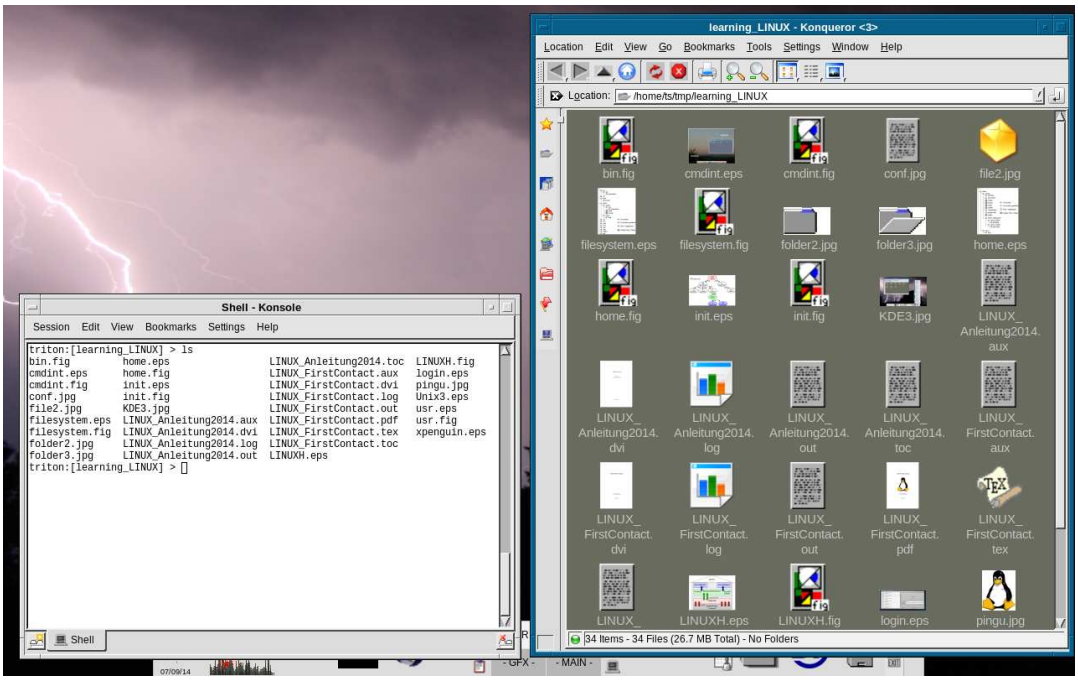

Abbildung 9: *Alle Dateien können sowohl mit der Konsole (links) wie auch mit dem Desktop-Datei-Manager (rechts) angezeigt bzw. aufgerufen werden. Beide Fenster zeigen die gleichen Dateien an!*

Dies erzeugt eine Ausgabe der Art:

```
[~]> /home/schanz
```

Das ’Kommandoprompt’ zeigt in diesem Fall also in das HOME-Verzeichnis von User ’schanz’!
Die Bezeichnung 'root'-Verzeichnis ist kontextabhängig, nicht verwirren lassen! Das 'root'-Verzeichnis des Systems, also '/', ist etwas anderes als das 'root'-Verzeichnis eines Users, zum Beispiel User 'schanz'. Das root-Verzeichnis von User 'schanz' liegt unter '/home/schanz/', es ist eben das 'Wurzel'- Verzeichnis von 'schanz' und nicht das vom System! Das ’root’-Verzeichnis von Users wird deshalb als User-HOME-Verzeichnis bezeichnet. Zu allem Überfluß wird der SuperUser unter LINUX ebenfalls als 'root' bezeichnet. Das ist historisch bedingt und liegt daran, dass das User-HOME-Verzeichnis des SuperUsers früher das 'root'-Verzeichnis des Systems war, also alle User-Dateien des SuperUsers direkt unter '/' gespeichert wurden. Heute hat der SuperUser bzw. der Root-User sein eigenes User-HOME-Verzeichnis das unter '/root' liegt. Alles klar?

Die für LINUX wichtigsten Verzeichnisse im ’root’-Verzeichnis sind:

```
/usr    : enthaelt das Betriebssystem und die Open-Source Applikationen
/etc    : enthaelt die Konfiguration des Systems, von Services und Applikationen
/boot   : enthaelt den KERNEL und den Bootloader
/home   : enthaelt die User-HOME-Verzeichnisse
/var    : enthaelt die log-Dateien des Systems und von Applikationen
/dev    : enthaelt die Devicetreiber fuer den KERNEL
```

```
/root   : enthaelt die User-Dateien des Root-Users (SuperUsers)
/tmp    : enthaelt temporaere Dateien die das System anlegt
/bin    : enthaelt das Core-System
/sbin   : enthaelt Applikationen fuer Devices (z.B. Filessystemoperationen)
/mnt    : enthaelt montierte Plattendateisysteme
/media  : enthaelt montierte Dateisysteme von portablen Geraeten (USB-Sticks usw.),
          die meistens automatisch ueber automounter montiert werden.
/opt    : enthaelt kommerzielle Applikationen
```

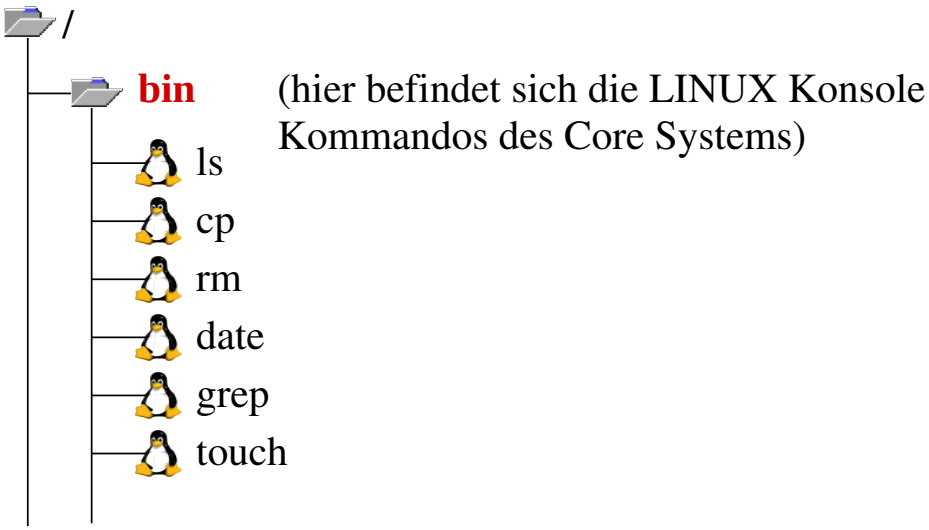

Abbildung 10: *Das '/bin' Verzeichnis enthält die LINUX-Konsole Kommandos die zum Kernsystem gehören.*

Der größte Teil der LINUX-Konsole-Kommandos befindet sich entweder unter '/bin' (Core-Kommandos) oder unter '/usr/bin' wenn es sich um Erweiterungen oder größere Dienstprogramme handelt, vergleiche hierzu auch Abbildung 10 und 11. Der Unterschied zwischen Applikation und Kommando ist ohnehin fließend, im Grunde sind auch Kommandos Applikationen.

```
Computer sind unzuverlässig,
   aber Menschen sind noch
   unzuverlässiger. Jedes
System, daß von menschlicher
Zuverlässigkeit abhängt, ist
   unzuverlässig. - Gilb
```

Im Verzeichnis '/usr' liegt der größte Teil der LINUX-Betriebssystems. Das sind neben System-Kommandos **(bin)** auch Anwendersoftware, Funktionsbibliotheken **(lib)**, Dateien zur Softwareentwicklung **(include)** und die Dokumentation dieser Software **(man)**. Bei den Funktionsbibliotheken handelt es sich um sogenannte 'shared libraries' (shared

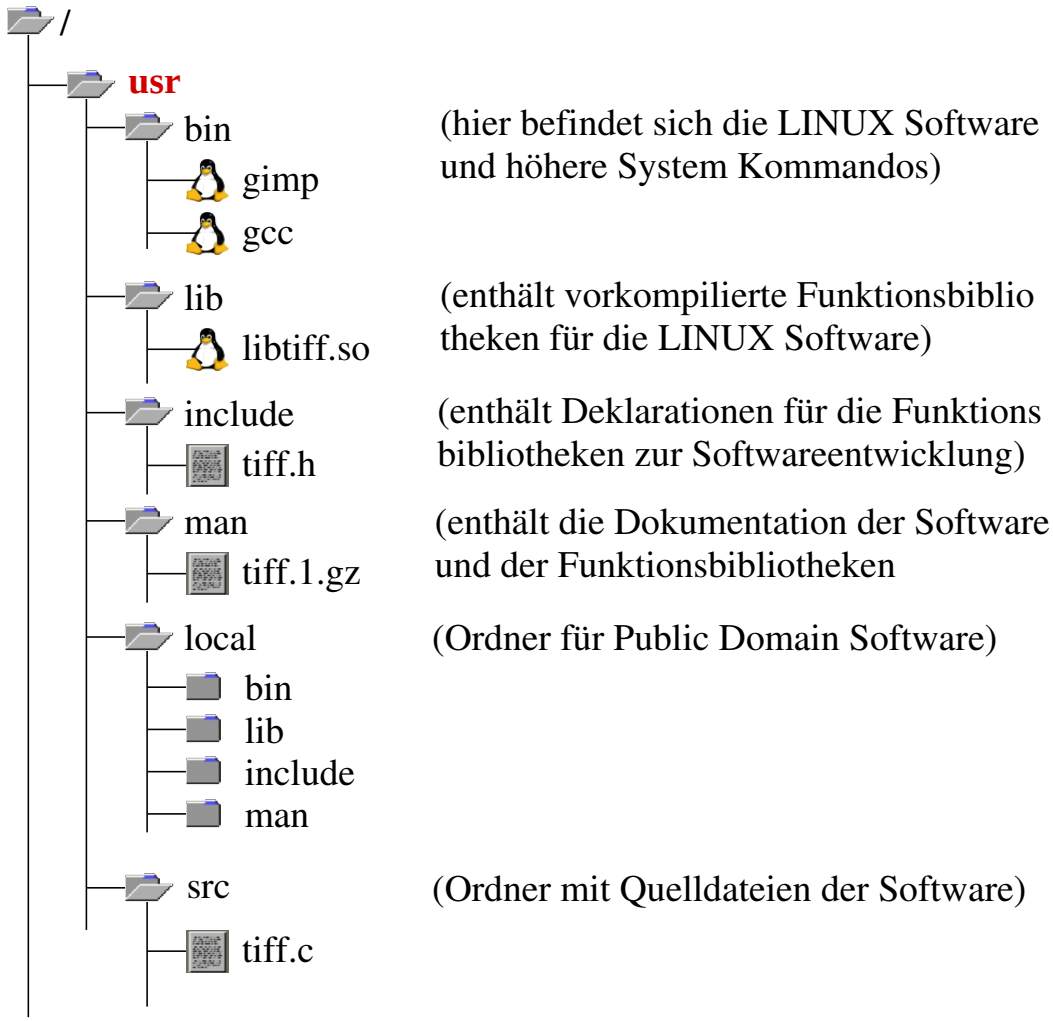

Abbildung 11: *Das '/usr' Verzeichnis enthält den größten Teil der Software des LINUX-Betriebssystems.*

objects (.so)). Das sind Bibliotheken von Funktionsaufrufen, die auf LINUX installiert werden können, und die von einer Vielzahl von Programmen gemeinsam genutzt werden. So kann Software kompakt und einfach gehalten werden, zur Zeit ihrer Ausführung verwendet die Software dann einfach die unter '/usr/lib' gespeicherten Funktionsaufrufe mit. Der Nachteil ist, wenn sie aus '/usr/lib' eine dieser '.so'-Bibliotheken entfernen oder upgraden ist davon mehr als eine einzige Software auf Ihrem System betroffen! Typische LINUX-Systeme besitzen zwischen ca. 500 und 1500 solcher '.so' Bibliotheken unter '/usr/lib'.

Die Untergliederung der Software auf die Verzeichnisse 'bin', 'lib', 'include' und 'man' ist typisch für LINUX. Sie werden diese Gliederung vielleicht auch in anderen Verzeichnissen als '/usr' finden.

9 Das User-HOME-Verzeichnis

Unter LINUX hat jeder Benutzer sein eigenes User-HOME-Verzeichnis wie es in Abbildung 12 dargestellt ist. Das Verzeichnis hat den Namen des Users und liegt unter '/home/<username>/'. Das User-HOME-Verzeichnis eines Users ist normalerweise per Default gegen andere User gesichert → siehe Dateirechte. Kein User kann die Dateien eines anderen Users einsehen, löschen oder aufrufen. Der Systemadministrator ist der einzige, der (im Prinzip - wird aber nicht - Ehrenkodex der Admins ;-)) Ihre Dateien ansehen kann.

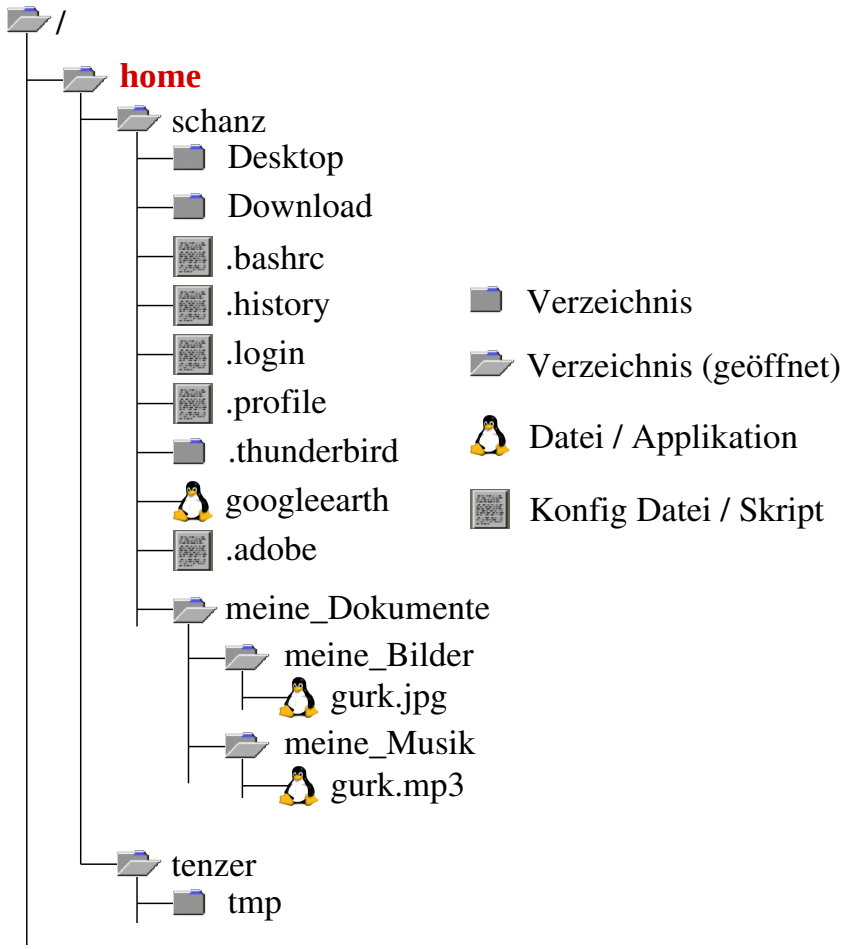

Abbildung 12: *Das User-HOME-Verzeichnis beherbergt alle Dateien und Verzeichnisse des betreffenden Users inklusive aller Konfigurationsdateien für die Software, die der User eingerichtet hat. Diese Konfigurationsdateien beginnen mit einem Punkt '.' und enthalten die privaten Einstellungen, die der User in der betreffenden Software vorgenommen hat. Der User ist frei darin unterhalb seines User-HOME-Verzeichnisses so viele Ordner und Dateien anzulegen wie er will, soweit die System-Ressourcen das zulassen.*

Jeder User kann unterhalb seines eigenen User-HOME-Verzeichnisses (also z.B. '/home/schanz/') machen was er will, d.h. z.B. so viele Unterverzeichnisse anlegen oder so viele Daten speichern wie er will, zumindest solange das Dateisystem noch Ressourcen

frei hat und keine Quotas verwendet. LINUX-Accounts an Universitäten oder Schulen verwenden meistens Quotas und beschränken auf diese Weise den Speicher den ein User auf dem System belegen darf. Quotas schützen andere User davor, dass ein einziger User den ganzen Speicherplatz des Systems für sich belegt.

Das HOME-Verzeichnis des Users enthält auch lokale Konfigurationsdateien von Programmen, die der User eingerichtet hat. Auf diese weise können Einstellungen von Programmen für jeden User getrennt gespeichert werden (private Einstellungen). Jeder User ruft Programme (die meisten davon unter '/usr/bin') mit seinen eigenen privaten Parametern auf. Diese Parameter sind in den User-Konfigurationsdateien in seinem User-HOME-Verzeichnis gespeichert. Die Config-Dateien heissen oft wie die Programme, fangen aber mit einem '.' im Dateinamen an.

10 Such-Pfad

Eine häufige von LINUX-Anfängern gestellte Frage ist: „Wo finde ich denn die Programme?“

Das ist wahrlich keine einfache Frage. Wo Programme bei LINUX hin installiert werden ist sehr unterschiedlich und abhängig von der Distribution, vom Urheber oder Hersteller des Programms bzw. den Vorlieben des Administrators. Es gibt keinen einen Ort, wo ausführbare Programme hin installiert werden. Das ist eine der Schwächen von LINUX gegenüber UNIX. Das Einzige was üblicherweise einheitlich ist, ist das Programme in einem Verzeichnis liegen das den Namen 'bin' hat, also z.B. '/bin', '/usr/bin', /usr/local/bin' usw.

Bei UNIX-Systemen gehören die Programme entweder zum Betriebssystem und liegen im Verzeichnis „/usr/bin“ des Dateisystems, oder es sind kommerzielle Applikationen die gewöhnlich ein eigenes Verzeichnis unter „/opt/“ des Dateisystems einnehmen, z.B.:

```
Oracle Datenbank:                   /opt/oracle/bin
Adobe Reader:                       /opt/adobe/bin
Xilinx FPGA Development system:     /opt/xilinx/bin
... usw.
```

Bei LINUX ist leider alles etwas chaotisch. Betriebssystembefehle liegen in der Regel ebenfalls unter „/usr/bin“ des Dateisystems, wie bei UNIX. Opensource-Applikationen werden inzwischen allerdings zum Teil auch nach „/usr/bin/“ installiert, wo sie eigentlich nichts zu suchen haben. Manche Distributionen installieren ihre Applikationen aber auch nach „/usr/local/bin/“ oder nach /usr/local/X11/bin/, andere nach „/usr/X11R6/bin/“, wieder andere nach „/usr/share/bin/“, usw. Der Internetbrowser 'firefox' z.B. installiert sich sinnigerweise nach „/usr/lib64/“, warum wissen die Entwickler alleine!? Ist das Installationsverzeichnis nicht im Suchpfad (echo $PATH), lässt sich das Programm nach der Installation aus der Konsole nicht ohne 'absolute' Pfadangabe aufrufen (... command not found ...). Es gibt leider keine einfache Lösung für dieses Problem, 'find' ist dein Freund ;-) .

Suchen Sie das Verzeichnis in das das Programm installiert wurde das Sie ausführen möchten und nehmen Sie das betreffende Verzeichnis in den Such-Pfad auf, indem Sie die Ressource-Datei '.bashrc' (z.B. '/home/schanz/.bashrc') in ihrem User-HOME-Verzeichnis editieren.

Die Konsole definiert den Such-Pfad über eine Umgebungsvariable die 'PATH' heißt und alle Verzeichnisse des Such-Pfads enthält. Ist der Such-Pfad erweitert, reicht es den Programm- oder Befehlsnamen, den man ausführen möchte, ohne weitere Pfadangabe in die Konsole einzugeben und das Programm oder Kommando wird ausgeführt. Das ist sehr hilfreich, weil Kommandos und Programme in vielen unterschiedlichen Verzeichnissen des Dateisystems gespeichert sein können. Durch die PATH-Umgebungsvariable brauchen Sie nicht jedes mal zu wissen wo das Kommando oder das Programm liegt, Sie geben den Namen des Programms einfach ein (ohne den absoluten Pfad) und los gehts. Die PATH-Variable ist üblicherweise in der Ressource-Datei der Shell in Ihrem User-HOME-Verzeichnis definiert, also z.B. '.bashrc' definiert.

Die Zeile mit dem PATH-Eintrag in '.bashrc' sieht in etwa so aus:

```
PATH=/bin:/usr/bin:/usr/bin/X11:/usr/local/bin:/usr/local/bin/X11:/home/schanz/.dt/bin
:$PATH
```

Angenommen Sie wollen den Suchpfad um das Verzeichnis mit den ausführbaren Dateien der NASA HEASOFT-Software erweitern, dann hängen Sie das entsprechende Verzeichnis '/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/bin' einfach an die Liste der PATH-Variable wie folgt an:

```
PATH=/bin:/usr/bin:/usr/bin/X11:/usr/local/bin:/usr/local/bin/X11:/home/schanz/.dt/bin
:/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/bin:$PATH
```

Die einzelnen Verzeichnisse sind mit Doppelpunkten ':' aneinander gehängt. Die Liste gibt an, wo und in welcher Reihenfolge die Konsole nach ausführbaren Programmen sucht. Sie können sich die PATH-Umgebungsvariable anzeigen lassen wenn Sie 'echo $PATH' in die Konsole eingeben. Nachdem PATH jetzt erweitert ist, enthält der neue Suchpfad die folgenden Verzeichnisse des Dateisystems:

```
/bin
/usr/bin
/usr/bin/X11
/usr/local/bin
/usr/local/bin/X11
/home/schanz/.dt/bin
/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/bin
```

Wenn Sie eine andere Shell als die Bourne-Shell verwenden, müssen Sie die zugehörige Ressource-Datei dieser anderen Shell bearbeiten und die PATH-Variable dort ändern. Sie wissen nicht welche Shell Sie verwenden? Das ist ebenfalls über eine Umgebungsvariable festgelegt, die 'SHELL' heißt. Geben Sie in die Konsole einfach 'echo $SHELL' ein und Sie sehen welche Shell die Konsole in ihrem Fall verwendet:

```
/bin/bash
```

Ein ähnlicher Mechanismus existiert für den Suchpfade der Man-Pages. Man-Pages sind Kurzanleitungen zu jedem Kommando oder Programm das auf dem System installiert ist. Sie können durch Eingabe von:

```
[~]> man <kommando/programmname>
```

die Ausgabe der Man-Page zu diesem Kommando starten. Versuchen Sie in der Konsole mal die Eingabe von:

```
[~]> man date
```

Damit erhalten Sie die Anleitung für das 'date'-Kommando, die Ausgabe sieht etwa so aus:

```
DATE(1)                          User Commands                          DATE(1)


NAME
        date - print or set the system date and time

 SYNOPSIS
        date [OPTION]... [+FORMAT]
        date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

 DESCRIPTION
        Display the current time in the given FORMAT, or set the system date.

        -d, --date=STRING
               display time described by STRING, not 'now'

        -f, --file=DATEFILE
               like --date once for each line of DATEFILE
 .........
```

Die Man-Pages folgen alle in etwa diesem Aufbau. Damit der Kommandointerpreter die betreffende Man-Page findet, gibt es eine Umgebungsvariable die 'MANPATH' heißt. Geben Sie in der Konsole ein:

```
[~]> echo $MANPATH
```

das erzeugt eine Ausgabe wie die hier:

```
/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/man:/usr/man:/usr/local/man:
/usr/share/man
```

Auch hier sind die Verzeichnisse in denen nach Man-Pages gesucht wird durch Doppelpunkte ':' getrennt. Man-Pages sind sehr hilfreich. Sie geben eine Kurzanleitung der Software, auch wenn Sie mal kein Internet zur Verfügung haben.

Die Steuerung von Software und Skripten mittels Umgebungsvariablen wie z.B. PATH oder MANPATH ist unter LINUX weit verbreitet. Sie können sich jederzeit eigene Umgebungsvariablen ausdenken und in Ihren Skripten verwenden. Alle auf Ihrem System definierten Umgebungsvariablen können Sie sich mit dem Befehl 'env' auf der Konsole anzeigen lassen. Versuchen Sie einfach mal die Eingabe:

```
[~]> env
```

11 LINUX Konfiguration

Das '/etc'-Verzeichnis enthält fast die gesamte globale Konfiguration des LINUX-Systems und der installierten Software. Das betrifft den Bootvorgang von LINUX genauso wie die Netzwerkkonfiguration, die Einstellungen des Fenstersystems (X11), die Secureshell (ssh), Filesystemeinstellungen, Firewall, Druckerkonfiguration, Daemons usw.

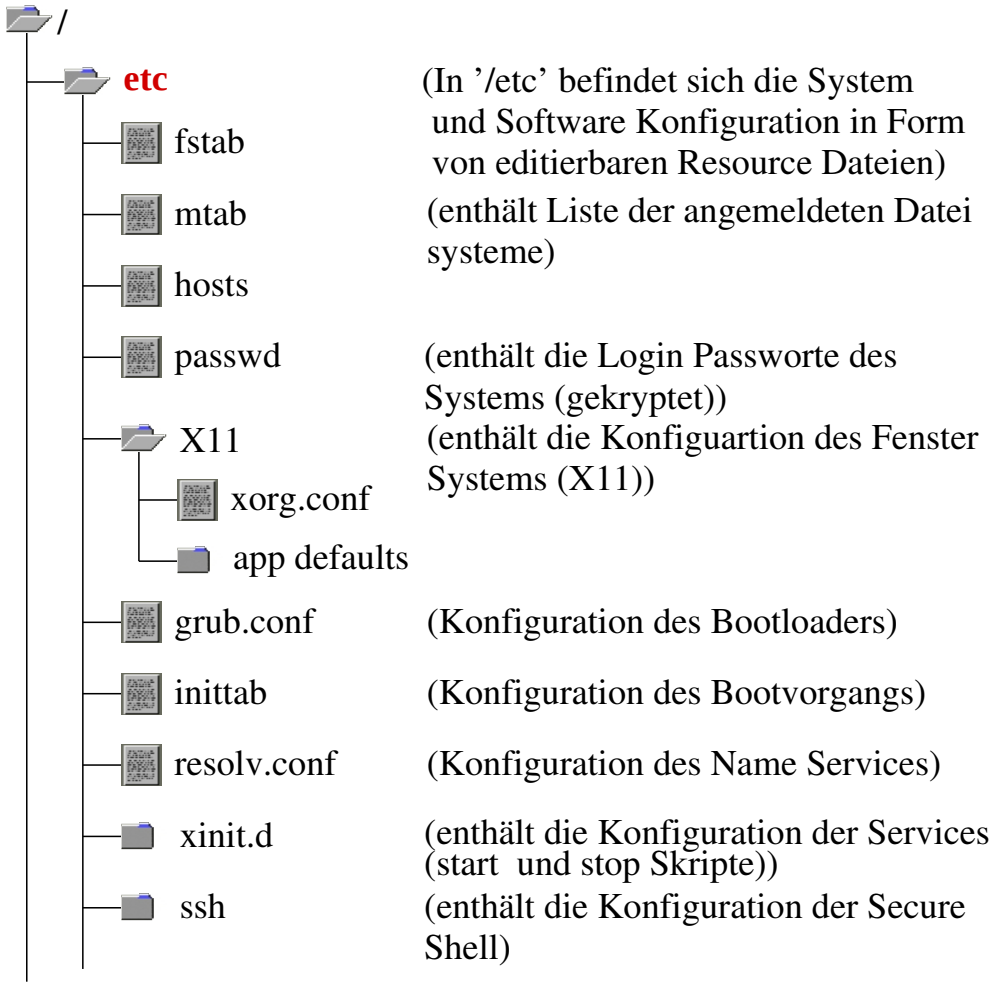

Abbildung 13: *Das '/etc' Verzeichnis enthält den größten Teil der System- und Softwarekonfiguration des LINUX Betriebssystems.*

Anders als WINDOWS verwendet LINUX keine Datenbank (registry) für die Konfiguration, sondern unzählige editierbare Konfigurationsdateien (Ressource-Dateien). Der Vorteil liegt darin, dass die Konfiguration verschiedener Software oder Dienste voneinander unabhängig ist. Das erhöht die Stabilität und die Sicherheit des Systems, das nicht von einem einzigen Mechanismus abhängt. Ein Fehler in der Konfiguration eines Dienstes oder einer Software hat keinen Einfluß auf die Konfiguration eines anderen Dienstes oder einer anderen Software. Bei einer Datenbankorientierten Konfiguration beeinflusst die Konfiguration eines einzelnen Dienstes unter Umständen das gesamte System, spätestens

nach der Installation neuer Software, wenn das System neu gebootet werden muss damit die Datenbank neu eingelesen wird. Abbildung 13 illustriert den Aufbau des '/etc'-Verzeichnisses mit ein paar wichtigen Konfigurationsdateien des Systems.

Um irgend eine der Konfigurationsdateien unter '/etc' zu verändern benötigen Sie Super-User Privilegien. Dies erfordert in der Regel Expertenwissen und ist für Einsteiger nicht zu empfehlen. Besser Sie halten sich am Anfang an eines der grafischen Konfigurationsprogramme, die der Desktop zur Verfügung stellt um Systemeinstellungen zu konfigurieren oder zu verändern.

12 Konsole-Kommandos (most wanted)

Dieses Kapitel gibt Ihnen einen kurzen Überblick über die wichtigsten Konsole-Kommandos des LINUX-Betriebssystems. Die Liste hier ist bei weitem nicht vollständig! Die Befehle von LINUX stehen im Dateisystem unter '/bin'. Mit dem Konsole-Kommando 'ls' können Sie sich die Liste der Kommandos anzeigen lassen. Bedenken Sie aber, dass zahlreiche Kommandos auch unter '/sbin' oder '/usr/bin' liegen. Nur ein kleiner Teil (der wichtigste) dieser Kommandos wird in dieser Anleitung behandelt, die meisten Kommandos besitzen zudem zahlreiche Optionen! Wie bereits erwähnt, gibt es zu jedem Befehl mit 'man <Befehl>' eine Kurzanleitung. In der Regel sind die hier aufgeführten Konsole-Kommandos alle im Such-Pfad. Sie können die Befehle alle ohne Pfadangabe direkt in die Konsole eingeben:

```
Am besten man verwirrt immer
  nur einen Punkt nach dem
   anderen. - Kernighan &
           Ritchie
```

12.1 ls

Das **'ls'**-Kommando ('list') erzeugt eine sortierte Ausgabe der Dateien die sich im aktuellen Verzeichnis befinden. Die Art der Sortierung kann über zahlreiche Optionen eingestellt werden. Beispiele:

```
ls        : list           : listet die Namen der Dateien oder Verzeichnisse
ls -l     : list long      : zusaetzlich Dateirechte, Dateigroesse, Erzeugungsdatum usw.
ls -al    : list all long  : zusaetzlich auch 'verdeckte' Dateien, die mit '.' anfangen
ls -ils   : list inodes    : zusaetzlich werden 'inodes' (hardlinks) angezeigt
ls -altr  : list all long timesort reverse : zusaetzlich zeitsortierte Ausgabe
```

Das 'ls -l'-Kommando erzeugt eine Ausgabe wie z.B.:

```
total 8128
drwxr-xr-x. 2 schanz users    4096 Mar 30  2017 Desktop
drwxr-xr-x. 2 schanz users    4096 May 24 10:01 Docs
drwxr-xr-x. 5 schanz users    4096 Aug  3 17:56 Downloads
drwx------  4 schanz users    4096 Aug 23 16:09 Dropbox
drwxr-xr-x  2 schanz users    4096 Sep 15  2017 Mount
```

```
-rw-r--r--  1 schanz users  590946 Aug 26 10:54 Screenshot from 2019-08-26 10-54-29.png
-rw-r--r--  1 schanz users 7683180 Aug 26 11:03 cat.jpg
lrwxrwxr-x  3 schanz users    4096 Jul 15 14:53 gurk -> scan
drwx------  2 schanz users    4096 May  1  2016 mail
drwxr-xr-x  2 schanz users    4096 Aug 27 12:35 scan
drwxr-xr-x  2 schanz users    4096 Jul 15 14:53 tmp
......
|    |      |    |     |      |    |       |     |
|  Rechte links owner group Bytes Datum   Zeit  Name
| (Access
|  Flags)
|
d = directory -> tmp ist ein Verzeichnis, alles andere sind Dateien
l = link -> gurk ist ein Link (Zeiger), der auf das Verzeichnis 'tmp' zeigt.
```

Die Ausgabe oben zeigt den typischen Inhalt eines User-HOME-Verzeichnisses. Es enthält einige Dateien, die der User angelegt hat und eine Reihe von Verzeichnissen, die zu jedem Login gehören. Wenn Sie 'ls' mit der Option '-l' aufrufen, bekommen Sie auch die Dateirechte der jeweiligen Dateien und Verzeichnisse mitangezeigt. Diese enthalten neben den 'Access-Flags', den Besitzer ('owner'), die Gruppe ('group'), der die Datei angehört, die Dateigröße in 'Bytes', 'Datum' und 'Zeit'punkt wann die Datei angelegt wurde und den 'Namen' der Datei.

12.2 pwd

Das **'pwd'**-Kommando ('print working directory') zeigt Ihnen an, an welcher Stelle des Dateisystems Sie sich befinden. Die Eingabe von:

```
[~]> pwd
```

erzeugt eine Ausgabe ähnlich wie:

```
/home/schanz
```

und zeigt wo sich der Kommandointerpreter der Konsole im Moment befindet, hier in diesem Fall im User-HOME-Verzeichnis von User 'schanz'.

Wenn Sie die Orientierung verloren haben und nicht wissen wo im Dateisystem Sie sich befinden, dann geben Sie einfach den Befehl 'pwd' in die Kommandozeile ein.

Computer können nicht
wirklich denken. Leute denken
nur dass sie denken. (denken
wir.)

12.3 cd

Mit dem **'cd'**-Kommando ('change directory') können Sie das Verzeichnis in dem sich der Kommandointerpreter der Konsole befindet ändern, sich also innerhalb des Dateisystems bewegen. Wenn Sie den Kommando-Prompt in ein anderes Verzeichnis verlegen wollen dann geben Sie dieses hinter dem 'cd'-Kommando an, also z.B.:

```
cd /home/schanz  : wechselt in das User-HOME-Verzeichnis von User 'schanz'
cd /             : wechselt in das 'root'-Verzeichnis des Systems
cd /usr/bin      : wechselt in das '/usr/bin'-Verzeichnis
cd               : wechselt in ihr User-HOME-Verzeichnis
cd .             : wechselt in das aktuelle Verzeichnis (bleibt also dort wo es ist)
cd ..            : wechselt in das naechst hoeher liegende Verzeichnis des Dateisystems
cd -             : wechselt in das Verzeichnis vor dem letzten Aufruf von 'cd'
```

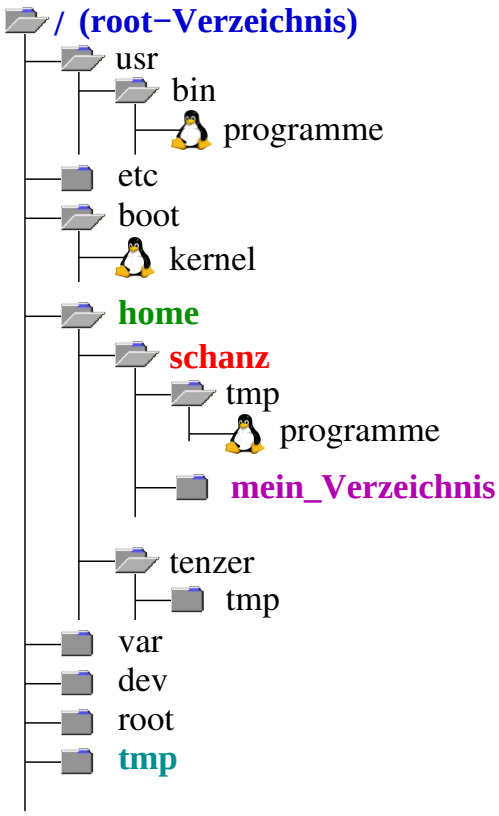

Abbildung 14: *Beispiel für die Navigation mit dem Befehl 'cd': Angenommen Sie sind User 'schanz'. Ihr User-HOME-Verzeichnis liegt dann folglich unter '/home/schanz/'. Mit der Eingabe von 'cd' in der Konsole gelangen Sie genau dort hin, nach '**/home/schanz**'. Geben Sie nun 'cd ..' ein, so gelangen Sie eine Verzeichnisebene höher, nach '**/home**', geben Sie nochmal 'cd ..' ein gelangen Sie noch eine Eben höher nach '**/**', ins '**root-Verzeichnis**' des Dateisystems. Geben Sie nun explizit 'cd /home/schanz/mein_Verzeichnis' ein und Sie gelangen nach '**mein_Verzeichnis**' das sich in Ihrem User-HOME-Verzeichnis befindet. Die Eingabe von 'cd /tmp' navigiert Sie weiter zu '**/tmp**', mit der Eingabe von 'cd -' gelangen Sie zurück nach '**mein_Verzeichnis**', in das Verzeichnis vor der letzten Eingabe von cd.*

12.4 cp

Mit dem **'cp'**-Kommando ('copy') können Dateien oder Verzeichnisse kopiert werden, zum Beispiel:

```
cp gurk.txt wurg.txt      : legt eine Kopie von 'gurk.txt' an, die 'wurg.txt' heisst.
cp gurk.txt /home/schanz/ : legt eine Kopie von 'gurk.txt' unter '/home/schanz' an, die
                           'gurk.txt' heisst
```

```
cp gurk.txt gurk.txt      : legt eine Kopie von 'gurk.txt' an, die 'gurk.txt' heisst.
                            Dieses Kommando schlaegt fehl, es kann keine zwei Dateien mit
                            demselben Namen im gleichen Verzeichnis geben!
cp gurk.txt /             : legt eine Kopie von 'gurk.txt' unter '/' an, die 'gurk.txt'
                            heisst. Die Kopie schlaegt vermutlich fehl, denn in '/' darf
                            nur der SuperUser schreiben.
cp -r tmp neu_tmp         : legt eine Kopie des Verzeichnisses 'tmp' an, die 'neu_tmp'
                            heisst. Der komplette Dateiinhalt wird mitkopiert. Die Option
                            '-r' im Aufruf steht fuer 'rekursiv' und kopiert alle Dateien
                            und alle Unterverzeichnisse mit.
```

Der 'cp'-Befehl legt eine Kopie der Datei oder des Verzeichnisses an. Die alte Datei bleibt unversehrt am Ursprungsort erhalten.

12.5 mv

Das **'mv'**-Kommando ('move') verschiebt Dateien oder Verzeichnisse innerhalb des Dateisystems oder benennt sie um:

```
mv gurk.txt wurg.txt          : verschiebt die Datei 'gurk.txt' in die Datei 'wurg.txt'.
                                Das kommt defakto dem Umbenennen der Datei gleich
                                (rename).
mv /tmp/gurk.txt /home/schanz/ : verschiebt die Datei 'gurk.txt' von '/tmp' in das Ver-
                                zeichnis '/home/schanz'. Der Name der Datei wird beibe-
                                halten, der Ort im Dateisystem geaendert.
mv tmp gurk                   : ist 'tmp' ein Verzeichnis, dann bewirkt 'mv' das Umbe-
                                nennen des Verzeichnisses, hier von tmp zu gurk
mv /home/schanz/gurk /tmp/    : ist 'gurk' ein Verzeichnis im User-HOME-Verzeichnis von
                                User 'schanz', dann bewirkt der Befehl hier das Ver-
                                schieben des gesamten Verzeichnisses samt Inhalt in das
                                Verzeichnis '/tmp' des Dateisystems.
```

Das 'mv'-Kommando verschiebt die Datei oder das Verzeichnis innerhalb des Dateisystems. Es gibt immer nur eine Instanz der Datei, die Ursprungsdatei wird gelöscht.

12.6 rm

Das **'rm'**-Kommando ('remove') löscht Dateien oder Verzeichnisse aus dem Dateisystem. Sie sollten mit 'rm' sehr vorsichtig sein. Die Löschung von Dateien oder Verzeichnissen mit 'rm' kann nicht mehr rückgängig gemacht werden, es gibt kein 'undo'!

```
rm gurk.txt             : loescht die Datei 'gurk.txt' aus dem aktuellen Verzeichnis
rm gurk.txt wurg.txt    : loescht die Dateien 'gurk.txt' und 'wurg.txt' aus dem Datei-
                          system
rm -r tmp               : loescht das Verzeichnis 'tmp' rekursiv (mit Inhalt) aus dem
                          Dateisystem
rm *                    : loescht alle Dateien aus dem aktuellen Verzeichnis des Datei-
                          systems. Diesen Befehl sollten Sie aus Sicherheitsgruenden so
                          nie eingeben!
                          Befinden Sie sich unbeabsichtigt im falschen Verzeichnis, so
                          loeschen Sie dort versehentlich den gesamten Inhalt!

                          Es ist besser Sie geben den Verzeichnisnamen im 'rm'-Befehl
                          mit an, also z.B. 'rm -r /home/schanz/tmp' oder 'rm /home/
                          schanz/tmp/*', wenn Sie nur den Inhalt von 'tmp' loeschen
                          wollen, aber nicht das Verzeichnis selbst.
rm /home/schanz/tmp/*   : loescht den Inhalt des Verzeichnisses '/home/schanz/tmp/'
```

12.7 echo

Der **'echo'**-Befehl gibt eine Zeichenkette oder den Inhalt einer Variablen aus:

```
echo "Hello World"        : gibt die Zeichenkette "Hello World" auf der Konsole aus.
echo $PATH                : gibt den Inhalt der Umgebungsvariablen PATH aus.
```

Der 'echo'-Befehl wird meistens verwendet um eine Textausgabe in Skripten auf der Konsole zu erzeugen.

12.8 cat

Der **'cat'**-Befehl ('concatenate') gibt den Inhalt einer Datei auf 'stdio' (Standard I/O) aus. Beispiel:

```
[~]> cat .bashrc
```

gibt den Inhalt der Datei '.bashrc' auf 'stdio' in der Konsole aus.

```
Ein Elefant ist eine Maus mit
       Betriebssystem
```

12.9 grep

Mit dem **'grep'**-Befehl können Sie nach Zeichenketten (Strings) in 'lesbaren' Dateien suchen lassen. Angenommen Sie möchten sich alle Dateien in Ihrem User-HOME-Verzeichnis anzeigen lassen die ihren Login-Namen (z.B.'schanz') enthalten. User 'schanz' könnte hierzu beispielsweise folgendes eingeben:

```
[~]> grep schanz *
```

Der Stern '*' bewirkt hier, dass 'grep' jede Datei des aktuellen Verzeichnisses öffnet und darin nach dem String 'schanz' sucht. Die Ausgabe von 'grep' gibt Ihnen den Dateinamen in dem 'grep' fündig wurde zusammen mit der Zeile in der Datei, die den String enthält auf der Konsole aus.

Möchten Sie beispielsweise wissen, ob ein spezieller User-Account (z.B. 'root') auf Ihrem System existiert so geben Sie ein:

```
[~]> grep root /etc/passwd
```

Gibt es einen Eintrag für 'root' innerhalb der Datei '/etc/passwd', so erhalten Sie eine Ausgabe ähnlich wie:

```
root:x:0:0:root:/root/:/bin/bash
```

oder auf neueren LINUX-Systemen auf denen man sich per Default nicht mehr als 'root' anmelden darf eher etwas wie:

```
operator:x.11:0:operator:/root:/sbin/nologin
```

Versuchen Sie den letzten Aufruf mit Ihrem eigenen User-Account.

Im allgemeinen wird 'grep' dazu verwendet um nach einem speziellen Wort in einer Datei oder in einem Verzeichnis mit Dateien zu suchen.

12.10 more

Der **'more'**-Befehl gibt den Inhalt einer Datei seitenweise aus. Im Gegensatz zu 'cat' scrollt die Konsole nicht weg. Sie können die Ausgabe mit der 'Enter'- und der 'Space'-Taste fortsetzen. Beispiel:

```
[~]> more .bashrc
```

gibt den Inhalt der Datei '.bashrc' in der Konsole seitenweise aus.

Im allgemeinen wird 'more' als Textbrowser für Kommandos mit sehr umfangreicher Textausgabe verwendet.

12.11 less

Der **'less'**-Befehl gibt den Inhalt einer Datei seitenweise aus. Im Gegensatz zu 'cat' scrollt die Konsole nicht weg. Sie können die Ausgabe mit der 'Enter'- und der 'Space'-Taste fortsetzen und mit den Pfeiltasten in der Ausgabe navigieren. Beispiel:

```
[~]> less .bashrc
```

gibt den Inhalt der Datei '.bashrc' in der Konsole seitenweise aus.

Im allgemeinen wird 'less' als Textbrowser für Kommandos mit sehr umfangreicher Textausgabe verwendet.

12.12 tail

Das **'tail'**-Kommando gibt Ihnen die letzten n-Zeilen einer Textdatei aus. Das kann sehr hilfreich sein. Viele Textdateien sind sehr groß und ihre Ausgabe mit Befehlen wie 'cat' oder 'more' ist schwerfällig weil diese Befehle immer die gesamte Datei laden und von Beginn an anzeigen. Insbesondere bei Log-Dateien sind meistens nur die letzten paar Einträge von Interesse. Mit dem 'tail'-Kommando können Sie sich gezielt die 'n'-letzten Zeilen einer Datei anzeigen lassen. Geben Sie z.B. ein:

```
[~]> sudo tail -10 /var/log/messages
```

Obiger Befehl gibt die letzten 10 Zeilen der KERNEL-Log-Datei '/var/log/messages' aus. Diese Log-Datei wird mit jedem KERNEL-Systemaufruf länger, meistens interessiert man sich aber dafür, ob der letzte Aufruf erfolgreich war! Sie benötigen hier das 'sudo'-Kommando weil normale User auf '/var/log/messages' keinen Zugriff haben.

```
[~]> sudo tail -f /var/log/messages
```

Für ein ständiges mitlesen was sich in der Log-Datei '/var/log/messages' ändert kann das 'tail'- Kommando ebenfalls verwendet werden. Mit der Option '-f' zeigt 'tail' nur Änderungen der Datei an, d.h. jede neue Zeile die der Datei am Ende angehängt wird, wird von 'tail -f' ausgegeben. Das eignet sich hervorragend zum Mitlesen von Log-Dateien und wird auch meistens genau dafür eingesetzt.

12.13 touch

Mit dem **'touch'**-Kommando können Sie den Zeitstempel von Dateien aktualisieren oder leere Dateien anlegen. Das kann sinnvoll sein, weil manche Skripte das vorhandensein von Dateien testen.

```
[~]> touch gurk.txt
```

aktualisiert den Zeitstempel der Datei 'gurk.txt'. Falls 'gurk.txt' nicht existiert, wird die Datei mit der Größe 0 Byte neu angelegt.

Der 'touch'-Befehl 'berührt eine Datei nur. Das bedeutet der Inhalt der Datei wird von 'touch' nicht verändert, lediglich der Zeitstempel der Datei.

```
wisst ihr Typen was ihr tut,
  oder programmiert ihr nur
           herum?
```

12.14 mkdir

Mit **'mkdir'** ('make directory') legen Sie neue Verzeichnisse an. Gehen Sie z.B. in Ihr User-HOME-Verzeichnis (geben Sie in der Konsole 'cd' ein) und legen Sie dort ein Verzeichnis neu an mit:

```
[~]> mkdir mein_Verzeichnis
```

Es wird ein Verzeichnis mit dem Namen 'mein_Verzeichnis' in Ihrem User-HOME-Verzeichnis erzeugt. Angenommen Sie sind User 'schanz', dann wird also das Verzeichnis '/home/schanz/mein_Verzeichnis' damit angelegt.

Sie können auch eine ganze 'Hierarchie' von Verzeichnissen und Unterverzeichnissen auf einmal anlegen wenn Sie die '-p'-Option von 'mkdir' verwenden. Geben Sie dazu ein:

```
[~]> mkdir -p mein1/mein2/main3
```

Angenommen Sie sind User 'schanz' und befinden sich bei der Eingabe des obigen Kommandos in '/home/schanz/mein_Verzeichnis', dann wird dort entsprechend die neue Kette von Verzeichnissen angelegt: '/home/schanz/mein_Verzeichnis/mein1/mein2/mein3'. Hätten Sie sich zum Zeitpunkt der Eingabe in '/home/schanz' befunden, wäre das Ergebnis '/home/schanz/mein1/mein2/mein3'. Das Anlegen der neuen Verzeichnisse bezieht sich also 'relativ' auf die Stelle im Dateisystem an der das Kommando-Prompt steht, zum Zeitpunkt wenn Sie das Kommando eingeben.

12.15 rmdir

Mit **'rmdir'** ('remove directory') können Sie Verzeichnisse aus dem Dateisystem wieder entfernen. Wenn 'mein_Verzeichnis' das Verzeichnis ist, dass Sie entfernen wollen, dann geben Sie ein:

```
[~]> rmdir mein_Verzeichnis
```

oder besser, geben Sie den Pfad 'absolut' an, um jede Zweideutigkeit zu vermeiden und damit die Eingabe des Kommandos von jeder Stelle des Dateisystems aus ausgeführt werden kann:

```
[~]> rmdir /home/schanz/mein_Verzeichnis
```

Sie können anstatt 'rmdir' auch das Kommando 'rm -r' verwenden um Verzeichnisse aus dem Dateisystem zu löschen. Das Verzeichnis muss leer sein damit Sie es entfernen können. Wenn Sie ein Verzeichnis mit Inhalt entfernen wollen, dann geben Sie 'rm' mit der Option '-rf' ein, also z.B.:

```
[~]> rm -rf /home/schanz/mein_Verzeichnis
```

12.16 find

Mit dem **'find'**-Kommando können sie im Dateisystem nach Dateinamen oder Verzeichnisnamen suchen lassen. Der Befehl kann sehr vielseitig eingesetzt werden. Der 'find'-Befehl folgt dem Schema:

```
[~]> find <wo> <was> <wer> <wohin>
```

also **'wo'** im Dateisystem soll gesucht werden, **'was'** für eine Art soll die Suche sein, soll z.B. nach einem Namen oder einem Zeitstempel gesucht werden, **'wer'** wird gesucht, d.h. z.B. nach was für einem Namens-String wird gesucht und **'wohin'** erfolgt die Ausgabe? Ein paar Beispiele:

```
find . -name gurk -print  : sucht innerhalb und unterhalb des aktuellen Verzeichnisses
                            '.'; gesucht wird ein 'name' mit dem String 'gurk'. Die
                            Ausgabe erfolgt nach 'print', d.h. in die Konsole (Default).

                            Der Befehl sucht konkret nach dem Vorkommen von 'gurk'
                            innerhalb der Dateinamen aller Dateien im aktuellen Ver-
                            zeichniss und aller Unterverzeichnisse.
find /usr -name gurk      : sucht innerhalb und unterhalb des Verzeichnisses '/usr'
                            nach dem Dateinamen 'gurk'
find /usr -name '*gurk*'  : sucht innerhalb und unterhalb des Verzeichnisses '/usr'
                            nach Dateinamen die den string 'gurk' enthalten
find .                    : gibt eine Liste aller Dateien im aktuellen Verzeichnis
                            '.' und aller Unterverzeichnisse aus.
find . -type f            : findet nur Dateien und keine Verzeichnisse
find . -type d            : findet nur Verzeichnisse und keine Dateien
find . -type d -name a*   : findet alle Verzeichnisse die mit 'a' beginnen
find / -name "*.txt" -size +12000c : findet Dateien unter '/' und allen Unterver-
                                     zeichnissen, also im gesamten Dateisystem,
                                     deren Name auf '.txt' endet und die groesser
                                     als 12000 Byte sind.
find / -name core -exec rm -f '{}' \; : findet alle Dateien mit dem Namen 'core' im
                                        gesamten Dateisystem und uebergibt sie an
                                        das Kommando 'rm', also loescht diese.
```

Es gibt endlos viele Möglichkeiten 'find' einzusetzen und etwa 40 Suchkriterien die miteinander kombiniert werden können! Bitte schauen Sie in die Man-Page von 'find' für weitere Informationen.

was nicht im Computer ist,
existiert nicht

12.17 ps

Das **'ps'**-Kommando ('process status') gibt Ihnen eine Liste der laufenden Prozesse aus. Beispiel:

```
[~]> ps
```

erzeugt eine Liste der Prozesse die aus der aktuellen Konsole heraus gestartet wurden. Diese Liste entält die Prozess ID-Nummer (PID), das Terminal von wo aus der Prozess gestartet wurde, die von jeweiligen vom Prozess beanspruchte Systemzeit und den Namen des Prozesses:

```
  PID TTY          TIME CMD
 5684 pts/4    00:00:00 bash
 5709 pts/4    00:00:00 ps
```

Wenn Sie 'ps' mit der Option '-fe' eingegeben:

```
[~]> ps -fe
```

so erfolgt eine Ausgabe wie unter Kapitel 7. Es wird eine Liste aller auf dem System laufenden Prozesse ausgegeben.

12.18 kill

Mit dem **'kill'**-Kommando können Prozesse gewaltsam beendet werden, d.h. bevor das Programm zu ende abgelaufen ist. Es gibt mehrere Prioritäten für 'kill', siehe die Man-Page → 'man kill'. Gebräuchlich ist:

```
kill <PID>     : darin ist PID die Prozessnummer, die Sie vorher mit 'ps' ermittelt
                 haben. Bei Eingabe von 'kill' wird das betroffene Programm aufge-
                 fordert sich zu beenden. Es hat noch Zeit Daten auf Massenspeicher
                 herauszuschreiben und seine Arbeit ordnungsgemaess zu beenden.
kill -9 <PID>  : darin ist PID die Prozessnummer, die Sie vorher mit 'ps' ermittelt
                 haben. Bei Eingabe von 'kill -9' wird dem Programm keine Zeit mehr
                 gelassen seine Arbeit zu beenden, die beteiligten Prozesse werden
                 ohne Ruecksicht auf Verluste sofort beendet!
```

Das 'kill'-Kommando wird immer dann verwendet, wenn ein Prozess abgestürzt ist und nicht mehr reagiert oder wenn ein Prozess überdurchschnittlich viel CPU-Last verursacht und die Maschine dadurch zu langsam wird.

```
Programmierer sterben nicht,
  sie gehn verloren in der
       Prozessierung
```

12.19 chown

Mit dem **'chown'**-Kommando ('change owner') kann der 'owner' einer Datei geändert werden. Jede Datei unter LINUX hat einen Eigentümer ('owner') und gehört einer Gruppe ('group') an -> Siehe Dateirechte Kapitel 13. Mit den Befehlen 'chown', 'chgrp' und 'chmod' können diese Dateirechte geändert werden. Beispiel:

```
[~]> chown schanz gurk.txt
```

ändert den 'owner' der Datei mit dem Namen 'gurk.txt' auf den 'owner' 'schanz'.

12.20 chgrp

Mit dem **'chgrp'**-Kommando ('change group') kann die Gruppenzugehörigkeit einer Datei geändert werden. Jede Datei unter LINUX hat einen Eigentümer ('owner') und gehört einer Gruppe ('group') an -> Siehe Dateirechte Kapitel 13. Mit den Befehlen 'chown', 'chgrp' und 'chmod' können diese Dateirechte geändert werden. Beispiel:

```
[~]> chgrp users gurk.txt
```

ändert die Gruppenzugehörigkeit der Datei mit dem Namen 'gurk.txt' auf die Gruppe 'users'.

12.21 chmod

Mit dem **'chmod'**-Kommando ('change modus') können die Dateirechte einer Datei geändert werden. Jede Datei unter LINUX hat einen Eigentümer ('owner') und gehört einer Gruppe ('group') an -> Siehe Dateirechte Kapitel 13. Mit den Befehlen 'chown', 'chgrp' und 'chmod' können diese Dateirechte geändert werden. Beispiel:

```
[~]> chmod 744 gurk.txt
```

ändert die Dateirechte (Access-Flags) der Datei 'gurk.txt' auf den Modus '744' -> Für die Bedeutung siehe Kapitel 13 'Dateirechte'.

12.22 uname

Der Befehl **'uname'** gibt Ihnen Informationen über ihr System aus, z.B. Informationen über die KERNEL-Version:

```
[~]> uname -a
```

erzeugt eine Ausgabe wie:

```
Linux atlas.local.de 4.8.13-100.fc23.x86_64 #1 SMP Fri Dec 9 14:51:40 UTC 2016 x86_64
x86_64 x86_64 GNU/Linux
```

Hieraus erfahren Sie, dass es sich um ein LINUX-System mit dem Computernamen 'atlas' handelt, mit dem Kernel-Release 4.8.13-100, das am Fri Dec 9 14:51:40 UTC 2016 für Intels x86 64Bit-CPU-Architektur kompiliert wurde und SMP (Multiprozessorbetrieb) unterstützt.

12.23 man

Das Man-Page Kommando **'man'** haben wir schon kennen gelernt. Es gibt eine Kurzanleitung zu jedem LINUX- Konsole Befehl mit seinen möglichen Optionen aus. Ich empfehle von 'man' reichlich gebrauch zu machen, denn diese Anleitung hier gibt nicht im entferntesten die Optionen und damit die Möglichkeiten der hier vorgestellten LINUX-Kommandos wieder.

```
[~]> man man
```

erzeugt eine Ausgabe wie:

```
MAN(1)              Dienstprogramme fuer Handbuchseiten                  MAN(1)

BEZEICHNUNG
     man - eine Oberflaeche fuer die Online-Referenzhandbuecher

UEBERSICHT
     man  [-C Datei] [-d] [-D] [--warnings[=Warnungen]] [-R Kodierung] [-L
     Locale] [-m System[,...]] [-M Pfad] [-S Liste] [-e Erweiterung] [-i|-I] [
     --regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P Anzeigep
     rogramm] [-r Prompt] [-7] [-E  Kodierung]  [--no-hyphenation] [--no-justi
     fication] [-p Zeichenkette] [-t] [-T[Geraet]] [-H[Browser]][-X[dpi]] [-Z]
     [[Abschnitt] Seite ...] ...
     man -k [apropos Optionen] regulaerer_Ausdruck ...
     man -K [-w|-W] [-S Liste] [-i|-I] [--regex] [Abschnitt] Terminal ...
     man -f [whatis Optionen] Seite ...
     man -l [-C Datei] [-d] [-D] [--warnings[=Warnungen]] [-R Kodierung] [-L L
     ocale] [-P Anzeigeprogramm]  [-r  Prompt]  [-7] [-E Kodierung] [-p Zeiche
     nkette] [-t] [-T[Geraet]] [-H[Browser]] [-X[dpi]] [-Z] Datei ...
     man -w|-W [-C Datei] [-d] [-D] Seite ...
     man -c [-C Datei] [-d] [-D] Seite ...
     man [-?V]

BESCHREIBUNG
     man ist das System-Anzeigeprogramm fuer die Handbuchseiten. Jedes an man u
     ebergebene Argument Seite ist normalerweise der Name eines Programms oder
     einer Funktion. Gefunden und angezeigt wird die Handbuchseite, die auf  je
     des  der  Argumente passt.  Wenn  ein  Abschnitt  angegeben  wird, sucht m
     an nur in diesem Abschnitt der Handbuchseiten. Ohne eine explizite Angabe
     werden alle verfuegbaren Abschnitte in einer festgelegten Reihenfolge durc
     hsucht (per Vorgabe "1 1p 8 2 3 3p 4 5 6 7  9 0p n l p o 1x 2x 3x 4x 5x 6x
     7x 8xT", es sei denn, dies wird durch die SECTION-Anweisung in /etc/man_db
     .conf ausser Kraft gesetzt). Wenn die Seite in mehreren Abschnitten vorkom
     mt, wird nur die jeweils zuerst gefundene Seite angezeigt.
     .....
```

Mit der Option ’-k’ können Sie innerhalb aller Man-Pages nach einem Suchbegriff suchen lassen. Das ist oft sehr hilfreich, wenn Sie nicht genau wissen, zu welcher Man-Page der Suchbegriff passt. Beispiel: Sie suchen nach Man-Pages die sich mit Bandlaufwerken befassen, also das Wort ’tape’ beinhalten. Geben Sie hierzu ein:

```
[~]> man -k tape
```

Dies erzeugt die Ausgabe aller Man-Pages, die den Begriff ’tape’ enthalten und etwas mit dem Betrieb von Magnetbändern zu tun haben:

```
amanda-taperscan (7) - Amanda Taperscan Algorithms
amcheckdb (8)        - check Amanda database for tape consistency
amfetchdump (8)      - extract backup images from multiple Amanda tapes.
amflush (8)          - flush Amanda backup files from holding disk to tape
amlabel (8)          - label an Amanda tape
amrmtape (8)         - remove a tape from the Amanda database
amtape (8)           - Control Amanda changers
amtapetype (8)       - generate a tapetype definition by testing the device directly
smbtar (1)           - Shell script for backing up SMB/CIFS shares directly to UNIX tape
                       drives
st (4)               - SCSI tape device
tapelist (5)         - The list of Amanda volumes in use
tar (5)              - format of tape archive files
```

Viele Man-Pages sind leider nur in englischer Sprache verfügbar!

wir ertrinken in Information
aber hungern nach Erkenntnis
– John Naisbitt, Megatrends

12.24 df

Mit dem **'df'**-Kommando ('disk free') können Sie sich die aktuelle Belegung des Dateisystems anzeigen lassen und erfahren wieviel Speicherplatz noch frei ist:

```
[~]> df
```

erzeugt eine Ausgabe wie:

```
Filesystem              1K-blocks      Used Available Use% Mounted on
/dev/sda5                80431944  46051704  30294480  61% /
tmpfs                     1009936         0   1009936   0% /lib/init/rw
udev                        10240       884      9356   9% /dev
tmpfs                     1009936        12   1009924   1% /dev/shm
/dev/sda3                  198337     34907    153190  19% /boot
192.168.0.1:/mnt/dsk2/  3845710848 3476797440 173562880  96% /mnt/dsk2
```

In der Liste sind die Dateisysteme (Filesystem) mit ihrer jeweiligen Belegung und den sogenannten Mountpoints aufgeführt. In diesem Beispiel hier befindet sich unter '/' das LINUX- Dateisystem (Partition: /dev/sda5), '/boot' enthält den KERNEL (Partition: /dev /sda3). Das unter 'mnt/dsk2' montierte Dateisystem ist von einem Server (hier 192.-168.0.1) remote, d.h. über das Netzwerk hinzugemountet (Netzwerkfilesystem NFS) und liegt auf einem anderen Computer, kann aber benutzt werden als ob es lokal installiert wäre. Das NFS-Dateisystem ist zu 96% voll, beachten Sie aber auch die Gesamtgröße der Platte von etwa 4 TByte (3845710848 kByte).

12.25 top

Mit dem **'top'**-Kommando können Sie sich eine Liste der Prozesse mit der höchsten CPU Auslastung ausgeben lassen. Die Eingabe von:

```
[~]> top
```

in der Konsole erzeugt eine Ausgabe wie:

```
top - 14:35:23 up  6:08,  5 users,  load average: 0.12, 0.10, 0.18
Tasks: 223 total,   2 running, 221 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.3 us,  0.8 sy,  0.0 ni, 95.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1991500 total,   724244 free,   223524 used,  1043732 buff/cache
KiB Swap:  3905532 total,  3905532 free,        0 used.  1492352 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2334 schanz    20   0  588020  57776  42100 S   2.3  2.9   0:02.27 konsole
 2165 schanz     9 -11  454840  10632   9164 S   1.3  0.5   8:39.30 pulseaudio
 1688 schanz    20   0  304372  24296  15488 S   0.7  1.2   4:29.11 Xorg
 2370 schanz    20   0   55056   4216   3460 R   0.7  0.2   0:00.21 top
18120 schanz    20   0  587468  19064  15628 S   0.7  1.0   0:22.17 konsole
   21 root      20   0       0      0      0 S   0.3  0.0   0:02.36 rcuos/1
 8789 schanz    20   0  454800   2968   2864 S   0.3  0.1   0:01.74 sd_espeak
    1 root      20   0  128256   4676   3360 S   0.0  0.2   0:04.40 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.07 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   0:05.64 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:04.15 rcuos/0
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.04 migration/0
   12 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   13 root      rt   0       0      0      0 S   0.0  0.0   0:00.04 watchdog/0
```

Aus dieser Liste erfahren Sie z.B. dass die Auslastung der CPU (load) im Moment 0.12 (12 Prozent) beträgt, gerade 223 Prozesse laufen von denen zwei aktiv sind, während 221 inaktiv sind (sleeping). Die Maschine hat 2 GByte Memory (1991500 total), davon sind ca. 224 MByte belegt (223524 used) und ca. 1.04 GByte für Cache-Speicher verwendet. Etwa 700 MByte sind noch frei (724224 free). Die Maschine hat bis jetzt noch keine Dateien auf die Festplatte auslagern müssen, (Swap 0 used) usw. Die Liste darunter gibt die Prozesse mit dem höchsten Rechenbedarf (CPU-Last) aus. Die Angaben unter 'top' sind nur Momentaufnahmen und können sich sekündlich ändern.

```
        Das Ziel der
 Computerwissenschaft ist es
etwas zu bauen, daß zumindest
 so lange funktioniert bis es
         fertig ist.
```

12.26 reboot

Das **'reboot'**-Kommando dient dem Neustart des Computers. Es beendet das aktuell laufende LINUX-Betriebssystem und bootet die Maschine neu. Die Ausführung von 'reboot' erfordert SuperUser Privilegien und kann von Usern nicht ausgeführt werden -> 'reboot: Need to be root'. Siehe auch Kapitel 6.

12.27 shutdown

Das **'shutdown'**-Kommando dient dem Herunterfahren des Betriebssystems und dem Ausschalten des Computers. Die Ausführung von 'shutdown' erfordert SuperUser Privilegien und kann von Usern nicht ausgeführt werden -> 'shutdown: Need to be root'. Siehe auch Kapitel 6.

12.28 which

Mit dem Kommando **'which'** können Sie sich den Ort eines Kommandos oder Programms im Such-Pfad der Konsole anzeigen lassen. Beispiel:

```
[~]> which reboot
```

erzeugt die Ausgabe:

```
/usr/bin/reboot
```

Das Kommando 'reboot' liegt also im Verzeichnis '/usr/bin'. Auf diese Art können Sie herausfinden aus welchem Verzeichnis ein Kommando von der Konsole ausgeführt wird. Das kann wichtig sein, wenn Sie eine Software (vielleicht in verschiedenen Versionen) in verschiedenen Verzeichnissen gespeichert haben. Mit 'which' können Sie herausfinden, welche dieser Versionen tatsächlich verwendet wird.

12.29 ln

Mit dem **'ln'**-Kommando ('link') können Sie sogenannte Zeiger (Links) auf Dateien oder Verzeichnisse erzeugen. Das ist sehr hilfreich wenn Sie eine Datei oder ein Verzeichnis an mehreren Stellen des Dateisystems benötigen aber keine Kopien dieser Dateien oder Verzeichnisse anlegen wollen, die unnötig Speicherplatz verbrauchen würden. In einem solchen Fall können Sie an den Stellen des Dateisystems an denen die Datei oder das Verzeichnis auftauchen sollen einen Zeiger anlegen, der auf ein und dieselbe Datei bzw. Verzeichnis zeigt. Ein Beispiel: Sie sind Admin auf dem LINUX-System und Sie haben eine Bibliothek mit Zeitschriften im pdf-Format die sie einer Reihe von Benutzern auf Ihrem System anbieten wollen. Die Zeitschriften liegen im Dateisystem unter '/dsk2/Zeitschriften'. Mit dem Befehl 'ln' können Sie dieses Verzeichnis ihren Benutzern einfach und unkompliziert verfügbar machen, indem sie Links in den jeweiligen User-HOME-Verzeichnissen der User auf das Verzeichnis '/dsk2/Zeitschriften' anlegen. Konkret sieht das z.B. so aus:

```
[~]> ln -s /dsk2/Zeitschriften /home/schanz/Zeitschriften
[~]> ln -s /dsk2/Zeitschriften /home/tenzer/Zeitschriften
```

Dies erzeugt in den User-HOME-Verzeichnissen von User 'schanz' und User 'tenzer' jeweils ein Verzeichnis 'Zeitschriften', dass aber tatsächlich nicht unter '/home/schanz' bzw. '/home/tenzer' liegt sondern nur ein Zeiger auf das Verzeichnis 'Zeitschriften' unter '/dsk2/' ist. Wenn User 'schanz' in seinem User-HOME-Verzeichnis '/home/schanz' nun das Kommando 'ls -l' eingibt, so findet er den Eintrag:

```
lrwxrwxr-x 1 root root       21 2014-09-07 19:22 Zeitschriften -> /dsk2/Zeitschriften
```

User 'schanz' kann in seinem User-HOME-Verzeichnis einfach mit:

```
[~]> cd Zeitschriften
```

in dieses Verzeichnis wechseln, ganz so als würde es tatsächlich unter '/home/schanz' existieren. Das gleiche gilt natürlich auch für User 'tenzer' und für jeden anderen User für den sie einen solchen Link erstellt haben. In Wahrheit gibt es aber nur eine einzige Instanz des Verzeichnisses in Ihrem Dateisystem, eben '/dsk2/Zeitschriften', die Links erzeugen also nur lokale Abbildungen des Verzeichnisses ohne dass dies zusätzlich Speicherplatz verbraucht.

12.30 tar

Mit dem **'tar'**-Kommando ('tape archive reader') können Sie Archive von Dateien und Verzeichnissen erstellen oder expandieren. Eigentlich dient 'tar' ja dem schreiben und lesen von Bandlaufwerken, die Sie an ihren Computer anschließen können. Vor allem im professionellen Computerbereich wird 'tar' auch tatsächlich heute noch dafür eingesetzt, Server verwenden zum Grossteil auch heute noch Bandroboter um große Datenmengen zu sichern (backup). Im privaten Bereich sind Bandlaufwerke von USB-Sticks und Festplatten verdrängt worden. Der 'tar' Befehl ist unter LINUX aber sehr beliebt um Verzeichnisse oder Dateien zu bündeln, bevor sie meistens komprimiert werden. Solche 'tar'-Archive können sowohl Dateien wie auch Verzeichnisse enthalten. Es ist möglich

damit ganze Dateisysteme zu archivieren. Typischerweise kann man damit z.B. Softwareprojekte, Sammlungen von Bildern oder mp3-Dateien in eine einzige Datei verpacken, 'tar'-Archive eignen sich ausgezeichnet zur Lagerung oder zum Transport von Archiven. Angenommen Sie sind in Ihrem privaten User-HOME-Verzeichnis, dann erstellen Sie ein 'tar'-Archiv des Verzeichnisses 'tmp' z.B. wie folgt:

```
[~]> tar cvf tmp.tar tmp
```

In diesem Beispiel sei 'tmp' ein Verzeichnis mit Dateien, dass hier durch das 'tar'-Kommando in das Archiv 'tmp.tar' archiviert wird. Das Archiv 'tmp.tar' enthält schließlich das Verzeichnis 'tmp' mit all seinen Dateien und Unterverzeichnissen, alles archiviert in eine einzige Datei, nämlich 'tmp.tar'. Die Optionen 'cvf' bedeuten: 'c' für 'create', 'v' für 'verbose' und 'f' für 'file'. 'file' bedeutet hier das in eine Datei archiviert wird (eben tmp.tar) und nicht auf ein Bandlaufwerk-Device.

Wenn Sie eine solche Archiv-Datei wie 'tmp.tar' erhalten dann können Sie diese in ihren ursprünglichen Zustand expandieren durch Eingabe von:

```
[~]> tar xvf tmp.tar
```

Die Option 'x' in diesem Kommandoaufruf bedeutet, dass das Archiv 'expandiert' wird. Dabei wird die ursprüngliche Form und der Aufbau des Verzeichnisses 'tmp' mit all seinen Dateien und Unterverzeichnissen wieder hergestellt.

```
Die Frage ob Computer denken
 können ist so wie die Frage
ob U-Boote schwimmen können.
    - Edsger W. Dijkstra
```

12.31 gzip

Mit dem **'gzip'**-Kommando können Sie Dateien oder tar-Archive komprimieren. Die Komprimierung ist verlustlos, d.h. es gehen keine Daten verloren, sie bewirkt lediglich eine Verkleinerung der Dateigröße und verringert damit den Speicherbedarf der Datei im Dateisystem. Sie können mit 'gzip' aus unkomprimierten Dateien komprimierte erzeugen und aus komprimierten Dateien unkomprimierte, je nach Option die sie bei der Eingabe von 'gzip' aufrufen:

```
[~]> gzip -9 tmp.tar
```

In diesem Fall wird die Datei 'tmp.tar' komprimiert und die Datei 'tmp.tar.gz' erzeugt (oft einfach auch 'tmp.tgz' genannt), wobei 'tmp.tar.gz' die komprimierte, in der Speichergröße reduzierte, Datei ist.

Umgekehrt kann eine komprimierte Datei vom Typ '.gz' mit der Option '-d' wieder dekomprimiert werden, also z.B.:

```
[~]> gzip -d tmp.tar.gz
```

Dabei entsteht wieder die unkomprimierte Datei 'tmp.tar'.

12.32 bzip2

Mit dem **'bzip2'**-Kommando können Sie Dateien oder tar-Archive komprimieren. Die Komprimierung ist verlustlos, d.h. es gehen keine Daten verloren, sie bewirkt lediglich eine Verkleinerung der Dateigröße und verringert damit den Speicherbedarf der Datei im Dateisystem. Sie können mit 'bzip2' aus unkomprimierten Dateien komprimierte erzeugen und aus komprimierten Dateien unkomprimierte, je nach Option die sie bei der Eingabe von 'bzip2' aufrufen. 'bzip2' ist moderner als 'gzip' und komprimiert besser!

```
[~]> bzip2 -9 tmp.tar
```

In diesem Fall wird die Datei 'tmp.tar' komprimiert und die Datei 'tmp.tar.bz2' erzeugt (oft auch einfach 'tmp.tbz' genannt), wobei 'tmp.tar.bz2' die komprimierte, in der Speichergröße reduzierte, Datei ist.

Umgekehrt kann eine komprimierte Datei vom Typ '.bz2' mit der Option '-d' wieder dekomprimiert werden, also z.B.:

```
[~]> bzip2 -d tmp.tar.bz2
```

Dabei entsteht wieder die unkomprimierte Datei 'tmp.tar'.

12.33 date

Mit dem **'date'**-Kommando können Sie sich das Datum und die Uhrzeit, also die Systemzeit ausgeben lassen. Wenn Sie SuperUser Privilegien haben können Sie mit dem 'date'-Kommando auch die Systemzeit neu einstellen. Wenn Sie 'date' in die Konsole eingeben erhalten Sie als Ausgabe die aktuelle Systemzeit, z.B:

```
Wed Aug 28 17:02:01 CEST 2019
```

Mit dem Kommando **'cal'** können Sie auch eine Kalenderausgabe auf der Konsole erzeugen.

12.34 who, w, finger

Das **'who'**-Kommando zeigt Ihnen an, welche Benutzer (bzw. anderen Benutzer) auf Ihrem System angemeldet sind und von wo aus sich diese Benutzer angemeldet haben. Die Eingabe von 'who -a' erzeugt z.B. folgende Ausgabe:

```
           system boot  2014-07-08 08:51
           run-level 5  2014-07-08 08:51
LOGIN      tty3         2014-07-08 08:51              1855 id=3
LOGIN      tty2         2014-07-08 08:51              1853 id=2
LOGIN      tty6         2014-07-08 08:51              1861 id=6
LOGIN      tty4         2014-07-08 08:51              1857 id=4
LOGIN      tty5         2014-07-08 08:51              1859 id=5
a107     - tty7         2014-07-08 08:52  old         2046 (:0)
a107     + pts/0        2014-07-09 17:22  old         8389 (:0.0)
root     + pts/1        2014-09-09 10:35   .          5482 (triton.local.de)
schanz   + pts/2        2014-09-09 10:37   .          5514 (phobos.local.de)
           pts/3        2014-08-11 09:18             28188 id=ts/3  term=0 exit=0
           pts/4        2014-08-12 11:20              3678 id=ts/4  term=0 exit=0
```

Die Liste zeigt den 'login-Namen' des jeweiligen Users, die Terminalemulation mit der der User eingelogged ist (tty), den Zeitpunkt des login und den Anschluss von wo aus der User sich angemeldet hat.

Eine ähnliche Ausgabe wie 'who' erzeugt der Befehl **'w'** und der Befehl **'finger'**.

12.35 sudo

Das **'sudo'**-Kommando ('substitute user do') erlaubt es Ihnen die Identität eines anderen Users anzunehmen, vorausgesetzt sie besitzen die Legitimation, also das Passwort. Wenn Sie keinen User-Namen angeben erhalten Sie mit 'sudo <command>' zur Ausführung dieses einen Kommandos SuperUser Privilegien. Sie erhalten die Privilegien in der Konsole aber nicht dauerhaft. Für jedes Kommando und jede Aktion, die SuperUser-Rechte erfordert, müssen Sie 'sudo <command>' gefolgt vom SuperUser Passwort erneut eingeben. Wenn Sie z.B. eine Festplatte anmelden wollen, dann können Sie das wie folgt eingeben:

```
[~]> sudo mount /dev/sdb1 /mnt/dsk3
```

Das 'mount'-Kommando ist nur vom SuperUser ausführbar, deshalb benötigen Sie 'sudo' um das Kommando aufzurufen. Obiger Aufruf montiert die Festplatte unter dem Gerätenamen '/dev/sdb1' in das Verzeichnis '/mnt/dsk3' des Dateisystems.

Als SuperUser können Sie einzelne User gezielt von der Passwort-Eingabe befreien, wenn Sie diese User in die Liste '/etc/sudoers' eintragen. Entsprechende Vorsicht ist dabei geboten! Bedenken Sie, dass normale User vielleicht weder die Kenntnisse noch die Verantwortung haben, um mit SuperUser Privilegien richtig umzugehen!

12.36 su

Das **'su'**-Kommando ('substitute user') erlaubt es Ihnen die Identität eines anderen Users anzunehmen, vorausgesetzt sie besitzen die Legitimation, also das Passwort. Wenn Ihr System dafür eingerichtet ist, können Sie mit dem Kommando 'su -' die Identität des SuperUsers dauerhaft annehmen, was speziell für administratorische Aufgaben sehr hilfreich sein kann. Aktuelle LINUX- Systeme unterbinden meistens diese Funktion per Default, entmündigen also den User ein Stück weit. Sicher ist die Unterbindung von 'su -' in soweit verständlich, dass Sie als SuperUser aus Unwissen oder Unachtsamkeit auch immensen Schaden am System anrichten können, neuere LINUXe erlauben daher nur noch den Zugriff mittels 'sudo' auf SuperUser Privilegien. Sie können jedoch jedes System so umstellen, dass die Verwendung von 'su -' gestattet wird. Wenn Sie herausfinden können wie das gemacht wird, dann dürfen Sie 'su -' auch verwenden :-). Angenommen Sie sind User 'schanz', möchten aber die Identität von User 'tenzer' annehmen, dann geben Sie das 'su'-Kommando wie folgt ein:

```
[~]> su - tenzer
```

Das System fragt Sie dann nach dem User-Passwort von User 'tenzer'. Wenn das Passwort korrekt ist, nehmen Sie die Identität von User 'tenzer' an, ihr User-HOME-Verzeichnis ändert sich von '/home/schanz' zu '/home/tenzer' und es werden die Ressource-Dateien

in '/home/tenzer' ausgeführt. Seien Sie darauf gefasst, das User 'tenzer' seinen Arbeitsplatz anders konfiguriert hat als sie selbst und z.B. eine andere Terminalemulation, eine andere Shell oder einen anderen Editor verwendet als sie es gewohnt sind. Wenn Sie **'exit'** in die Konsole eintippen (oder **'ctrl d'** auf der Tastatur drücken) werden Sie abgemeldet und nehmen wieder die alte Identität (User 'schanz') an.

Mit der Eingabe von 'su' ohne Usernamen oder 'su -' ohne Usernamen können sie auf der Konsole dauerhaft SuperUser Rechte erhalten, es wird das Passwort der SuperUsers verlangt. Wenn Sie 'su -' verwenden sind sie nach erfolgreicher Anmeldung 'root'-User, ihr User-HOME-Verzeichnis liegt jetzt unter '/root', es werden die dort gespeicherten Config-Dateien verwendet. Beachten Sie das 'root' eine andere Shell haben kann als Sie es als User gewohnt sind. Als 'root'-User können Sie im Prinzip jede Datei des Systems öffnen, schreiben oder löschen. Sie sollten genau wissen was Sie tun, wenn sie als 'root'-User auf dem System Änderungen vornehmen. Das Ausspionieren von User-Dateien, lesen von User-eMails usw. ist ein absolutes NO GO! Die Zuwiderhandlung ist ein schwerwiegender Vertrauensbruch und wird weder in Unternehmen noch öffentlichen Institutionen geduldet. Sie werden Ihren Job als Administrator verlieren wenn Sie das versuchen!

Sie sollten nur solange 'root'-User bleiben wie das für die Arbeit notwendig ist. Wenn Sie **'exit'** in der Konsole eingeben oder **'ctrl d'** auf der Tastatur drücken, werden Sie als 'root'-User abgemeldet und kehren in den Status als normaler User zurück.

```
Das Problem mit Computern
ist, das Computer das machen
was Sie ihnen sagen und nicht
das was Sie wollen. - D.
Cohen
```

12.37 passwd

Mit dem **'passwd'**-Kommando ('passwort') können Sie das Passwort eines Users und natürlich ihr eigenes Passwort ändern. Die Eingabe von 'passwd' ohne Zusatz ändert das Passwort des eingeloggten Users, das System fragt zuerst nach dem alten Passwort und danach nach dem neuen Passwort.
Der Administrator kann für Passworte Mindestanforderungen und Ablaufzeiträume festlegen nach denen das Passwort automatisch erlischt. Das System fordert den User in diesen Fällen rechtzeitig auf ein neues Passwort zu erstellen.

12.38 env

Mit dem **'env'**-Kommando ('environment') können Sie sich die Umgebungsvariablen auf Ihrem System anzeigen lassen. Geben Sie in der Konsole das Kommando 'env' ein. Dies erzeugt eine Ausgabe wie:

```
MANPATH=/usr/man:/usr/contrib/X11R6/man:/usr/local/man:/net/usr/man:/opt/wabi/man:/opt
        /man:/usr/share/man
SSH_AGENT_PID=1817
HOSTNAME=atlas.local.de
XANBIN=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22
TERM=xterm-256color
SHELL=/bin/bash
HISTSIZE=500
PGPLOT_RGB=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/lib/rgb.txt
PGPLOT_FONT=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/lib/grfont.dat
HEADAS=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22
WINDOWID=54525957
FTOOLS=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22
LD_LIBRARY_PATH=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/lib
LHEASOFT=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22
USERNAME=ts
PAGER=/bin/more
PATH=/opt/heasoft-6.22.1/x86_64-unknown-linux-gnu-libc2.22/bin:/home/ts/.dt/bin:/bin:/
     usr/bin:/sbin:/usr/sbin:/usr/bin/X11:/usr/local/bin:/usr/local/bin/X11:/usr/contr
     ib/bin:/usr/contrib/bin/X11:/opt/Xilinx/14.7/ISE_DS/ISE/bin/lin64:/usr/dt/bin:/op
     t/fv5.4:/usr/local/bin
BROWSER_PATH=/bin/firefox
EDITOR=/bin/vi
LANG=De
....
```

Viele Einstellungen des Benutzerkontos werden von Umgebungsvariablen gesteuert. Sie können sich den Inhalt jeder Umgebungsvariablen auf der Konsole einzeln anzeigen lassen wenn Sie das Kommando 'echo $<Umgebungsvariable>' auf der Kommandozeile eingeben.

```
Es gibt nicht den geringsten
  Grund warum irgendjemand
 einen Computer für sich zu
hause brauchen sollte. - Ken
 Olsen (President of Digital
   Equipment Corporation),
   Convention of the World
 Future Society, in Boston,
            1977
```

12.39 ssh

Mit dem **'ssh'**-Kommando ('secure shell') können Sie sich auf Systemen die über Netzwerk an Ihr System angeschlossen bzw. über Netzwerk von Ihrem System aus erreichbar sind 'remote' anmelden. Damit das gelingt muss auf dem 'remote'-System ein ssh-Daemon laufen und Sie müssen auf dem 'remote'-System ein 'Login' besitzen. Angenommen ihr 'Login' auf dem 'remote'-System heißt 'schanz', der Name des 'remote'-Systems heißt 'demokrit', dann sieht die Anmeldung z.B. wie folgt aus:

```
[~]> ssh schanz@demokrit
```

Das 'remote'-System, in diesem Fall also 'demokrit' antwortet darauf mit der Passwortabfrage:

```
schanz@demokrit's password:
```

Hier geben Sie ihr Passwort ein, dass Sie auf dem 'remote'-System 'demokrit' verwenden (nicht das von der lokalen Maschine). Ist die Anmeldung erfolgreich, begrüßt Sie das 'remote'-System 'demokrit' mit:

```
Last login: Wed Mar 27 18:02:44 2013 from triton.local.de
demokrit:[~]>
```

Sie sind jetzt als User 'schanz' auf 'demokrit' eingeloggt und können das System via Konsole genauso wie den lokalen Rechner mit den LINUX-Konsole Kommandos steuern. Das Starten von Software die Fenster öffnet ist per 'ssh' nur möglich, wenn das System dafür eingerichtet ist und Sie beim Aufruf von 'ssh' die Option '-X', bzw. '-Y' (je nach System) mit angegeben haben, also z.B.:

```
[~]> ssh -X schanz@demokrit
```

Sie beenden die 'ssh'-Sitzung indem Sie das Kommando 'logout' oder 'exit' eingeben oder indem Sie einfach das Konsole-Fenster auf ihrem lokalen Rechner schließen.

Sollte Ihrem lokalen System der Name des 'remote'-Systems (in unserem Beispiel 'demokrit') nicht bekannt sein, so können Sie stattdessen auch die IP-Adresse von 'demokrit' (falls bekannt) angeben. Der Verbindungsaufbau sieht dann entsprechend z.B. so aus:

```
[~]> ssh -X schanz@192.168.0.215
```

Wenn Sie sich an Ihrem IAAT-Account der Uni-Tübingen von außerhalb via ssh anmelden wollen, dann geht das am einfachsten über die Domain 'astro.uni-tuebingen.de'. Für User 'schanz' sieht das dann z.B. so aus (versuchen Sie es mal mit Ihrem eigenen User-Account):

```
[~]> ssh -X schanz@astro.uni-tuebingen.de
```

Die Secure-Shell (ssh) stellt eine verschlüsselte Verbindung zu dem 'remote'-System her, die nicht abgehört bzw. mitgelesen werden kann. Die Übertragung per 'ssh' gilt als besonders sicher.

12.40 scp

Mit dem **'scp'**-Kommando ('secure copy') können Sie Dateien von einem 'remote'-System auf das eigene oder vom eigenen System auf ein 'remote'-Systeme kopieren. Um Dateien vom 'lokalen'-System auf ein 'remotes'-System zu kopieren geben Sie die zu kopierenden Dateien dem 'scp'-Kommando wie folgt an:

```
[~]> scp /home/schanz/tmp/* schanz@demokrit:/home/schanz/
```

Obiges Kommando kopiert alle Dateien aus dem Verzeichnis '/home/schanz/tmp/' auf das remote-System 'demokrit' in das Verzeichnis '/home/schanz/'. Das 'remote'-System 'demokrit' antwortet zunächst mit der Aufforderung das Passwort für User 'schanz' auf 'demokrit' einzugeben.

```
schanz@demokrit's password:
```

Wenn Sie nicht nur Dateien sondern auch Verzeichnisse kopieren wollen, müssen Sie beim Aufruf von 'scp' die Option '-r' (rekursiv) mit angeben. Beim Kopieren bleibt der Verzeichnisaufbau wie auf dem 'lokalen'-Rechner erhalten.

```
[~]> scp -r /home/schanz/tmp/* schanz@demokrit:/home/schanz/
```

Während des Kopiervorgangs wird eine Liste der kopierten Dateien und Verzeichnisse in der Konsole laufend angezeigt. Die Ausgabe sieht z.B. wie folgt aus:

```
filesystem.fig                              100% 6940     6.8KB/s   00:01
LINUX_FirstContact.aux                      100% 2979     2.9KB/s   00:00
LINUX_Anleitung2014.log                     100%   24KB  24.4KB/s   00:00
.....
```

Um Dateien von 'remote'-Systemen auf das 'lokale'-System zu kopieren geben Sie das 'scp'-Kommando wie folgt an:

```
[~]> scp -r schanz@demokrit:/home/schanz/tmp/* /home/schanz/
```

Obiges Kommando kopiert alle Dateien aus dem Verzeichnis '/home/schanz/tmp/' des 'remote'-Systems in das Verzeichnis '/home/schanz/' des 'lokalen-Systems', inklusive Verzeichnissen und Unterverzeichnissen. Beim Kopieren bleibt der Verzeichnisaufbau wie auf dem 'remote'-Rechner erhalten.

12.41 script

Vielleicht kann Ihnen das **'script'**-Kommando bei der Erkundungstour durch die Konsole hilfreich sein, wenn Sie z.B. diverse Konsole-Eingaben ausprobieren und später nachvollziehen wollen, was Sie alles getan haben. 'Script' loggt alle Ihre Eingaben in der Konsole mit. Sie starten die Aufzeichnung mit der Eingabe des Kommandos:

```
[~]> script
```

in der Konsole. Dadurch wird im Hintergrund ein Prozess gestartet der alle Ein- und Ausgaben der Konsole aufzeichnet und in eine Log-Datei mitschreibt. Sie beenden die Aufzeichnung mit dem Tastaturkürzel 'ctrl d'. Das 'script'-Kommando erzeugt die Log-Datei 'typescript' in dem Verzeichnis wo Sie 'script' gestartet haben. Diese Log-Datei enthält dann den gesamten Inhalt der kompletten Konsole-Session.

```
 LINUX wird Sie nicht daran
 hindern dumme Dinge zu tun,
denn sonst würde es Sie auch
 daran hindern schlaue Dinge
     zu tun. - Doug Gwyn
```

13 Dateirechte

Dateirechte regeln den Zugriff auf Dateien unter LINUX. Es existieren drei Gruppen, für die die Zugriffsrechte der Datei definiert sind:

OWNER GROUP OTHERS

Dateirechte können mit dem Befehl 'ls' angezeigt und mit den Befehlen **'chown'**, **'chgrp'** und **'chmod'** modifiziert werden. Der Befehl ls -al erzeugt zum Beispiel etwa folgende Ausgabe:

```
dr-xr-xr-x   2   root         sys       1024   Nov 17 14:02     bin
dr-xr-xr-x   7   root         sys       1024   Nov 28 23:12     config
-rw-r-xr--   1   schanz       users      400   Dec 12 00:10     gurk
|   |        |     |           |          |        |             |
| Rechte   Links  owner      group      Bytes    Datum          Name
|
Typ: d = Verzeichnis; l = Link; b/c = devices
```

Jede Datei und jedes Verzeichnis hat einen Eigentümer (owner) und gehört außerdem einer Gruppe (group) an. Zugriff auf die Datei kann der Eigentümer, die Gruppe, und alle anderen Benutzer haben. Dies wird durch Datei-Rechte geregelt, welche von den Access-Flags angezeigt werden.

```
 d   r w x   r w x   r w x      <- Access-Flags
 |     |       |       |
Typ   owner  group   alle anderen (rest)
```

Darin stehen die Buchstaben **'r'** für 'lesbar', **'w'** für 'schreibbar' und **'x'** für 'ausführbar'. Mit 'rwx' wird also für jede der drei Gruppen (owner/group/others) festgelegt was mit der Datei oder dem Verzeichnis erlaubt ist. Beispiel:

```
- rwx r-x r--  4  schanz  users  400  Dec 12 00:10  gurk
```

Für die Datei mit dem Namen 'gurk' gilt in diesem Fall hier:

```
owner: r w x :  der owner der Datei, User 'schanz', darf die Datei lesen, schreiben
                 und ausführen.
group: r - x :  Mitglieder der Gruppe 'users' dürfen die Datei lesen und ausführen,
                 können die Datei aber nicht schreiben, insbesondere nicht löschen!
rest: r - - :   Alle anderen User auf dem System können die Datei nur lesen, aber
                 nicht schreiben oder ausführen.
```

Die Access-Flags können mit dem Kommando 'chmod' geändert werden. Dabei gibt der numerische Wert hinter 'chmod' den neuen Modus an. Das Kommando:

```
[~]> chmod 744 gurk
```

ändert den Status der Datei 'gurk' zu:

```
- rwx r-- r--  4  schanz  users  400  Dec 12 00:10  gurk
```

Warum ändert die Eingabe von ’chmod 744 gurk’ die Access-Flags von ’gurk’ auf den neuen Modus ’rwxr--r--’?

Die Zahlen hinter dem Kommando ’chmod’ werden als Hexadezimalzahl betrachtet, die einzelnen Bits wirken dann wie Schalter und werden bitweise mit den Access-Flags ’rwx rwx rwx’ multipliziert. Dort wo das Bit eine ’1’ hat wird der Modus gesetzt, dort wo das Bit ’0’ ist steht ein ’-’. Beispiele:

```
hexadezimal  :   binaer  :    Modus
   0         :    000    :     ---
   4         :    100    :     r--
   5         :    101    :     r-x
   6         :    110    :     rw-
   7         :    111    :     rwx
```

Rufen Sie eine Konsole auf und wechseln Sie in das Verzeichnis ’/home’

```
[~]> cd /home
```

Geben Sie jetzt das Kommando ’ls -l’ ein. Sie erhalten eine Ausgabe ähnlich wie diese:

```
total 32
drwx------  51 a107   users 4096 Aug 29 11:09 a107
drwx------  33 cde    users 4096 Jul  9 17:09 cde
drwx------. 25 tenzer users 4096 May  2 18:28 tenzer
drwxrwxr--   2 prakt  ait   4096 Aug 12  2010 prakt
drwxr-xr-x   2 root   root  4096 Aug 30  2012 printer
drwxr-xr--  25 schanz users 4096 Jun 26 19:22 schanz
```

Sie sehen hier die User-HOME-Verzeichnisse auf dem System. Die meisten User in diesem Beispiel gehören der Gruppe ’users’ an. So wie es üblich ist, sind die Access-Flags der meisten User ’a107’, ’cde’, ’tenzer’ auf ’700’ gesetzt (Default). Die Dateien dieser drei User sind gegen Einsicht und gegen Zugriff aller anderen User geschützt, nur der jeweilige User selbst hat Zugriff auf sein eigenes Verzeichnis (rwx) und die Datein darin.

User ’schanz’ hat seine Access-Flags auf ’754’ geändert. Er erlaubt allen Mitgliedern der Gruppe ’users’, dass sie die Dateien in seinem User-HOME-Verzeichnis ’r’-lesen und ’x’-ausführen dürfen, der Schreibzugriff (’w’) ist verweigert. Anderen Usern des Systems, die nicht zur Gruppe ’users’ gehören, also dem Rest, erlaubt User ’schanz’ immerhin noch den ’Lesezugriff (’r’) auf die Dateien in seinem User-HOME-Verzeichnis, das ausführen und schreiben von Dateien ist für diese User nicht möglich. Jeder User bestimmt selbst über die Access-Flags seines Verzeichnisses, Sie können selbst entscheiden was andere User bei Ihnen dürfen und was nicht!

```
Mache niemals etwas einfach
und effizient wenn es einen
Weg gibt es auch kompliziert
  und wundervoll zu machen
```

Wie Sie sehen ist auch ’printer’ ein User auf dem System. Der ’owner’ seiner Dateien ist der SuperUser ’root’ und er gehört der ’group’ von ’root’ an. Der SuperUser darf die Dateien von ’printer’ lesen, schreiben und ausführen. Mitglieder der Gruppe ’root’ dürfen die Dateien nur lesen und ausführen, ebenso wie alle anderen User (Others) auf dem System die nicht zur Gruppe ’root’ gehören. Das ist die Bedeutung der Access-Flags ’755’ (rwx r-x r-x) für das HOME-Verzeichnis von User ’printer’.

14 Terminals

Die Konsole bietet Ihnen ein Kommandointerface zur Steuerung des Computers. In den frühen Jahren von UNIX hatten Computer noch keine Grafik und keinen Desktop, die Ein- und Ausgabe geschah zu dieser Zeit über sogenannte Terminals. Ein Terminal ist ein elektronisches Gerät mit einer Tastatur und einem Bildschirm, aber ohne eine eigene Recheneinheit oder einen eigenen Speicher. Solche Terminals wie in Abbildung 15 dargestellt waren an die Computer der damaligen Zeit angeschlossen (direkt oder über Netzwerk). Die User konnten sich von solchen Terminals aus an Ihrem Computersystem anmelden und mit dem Betriebssystem des Computers (UNIX) kommunizieren, Programme schreiben, starten, E-Mails verschicken usw., kurz alles was keine pixelbasierte Grafikausgabe erfordert. Das Terminal war das Interface zur Konsole des Computers (damals die einzige Möglichkeit mit dem Betriebssystem zu kommunizieren – ist heute bei Servern, Mainframe-Computern und Supercomputern zum Teil immer noch so).

Abbildung 15: *Terminal der Firma HP wie es zur Kommunikation mit Servern verwendet wird. Zu sehen ist die Ausgabe des 'top'-Kommandos auf der Konsole unter User 'schanz' auf dem UNIX-Server 'regulus'*

Was die Konsole betrifft, verwenden LINUX-System bis heute Terminals. Diese Terminals sind allerdings keine elektronischen Geräte mehr, sondern Emulationen, die auf der Konsole laufen, oder in gewissem Sinne die Konsole erzeugen. Das Fenster auf dem Sie ihre Kommandos eingeben ahmt ein solches Terminalgerät der damaligen Zeit nach. Natürlich gab es damals nicht nur einen Hersteller von Terminals und so gibt es heute unter

LINUX verschiedene Terminal-Emulationen, welche die Terminals der verschiedenen Hersteller von damals simulieren. Hinzugekommen sind inzwischen auch generalisierte Terminals, die ihren Ursprung nicht direkt in der Hardware der Vergangenheit haben. Jede Distribution und jeder Desktop-Manager bringt seine eigene Terminal-Emulation mit. Zum Glück sind die Grundfunktionen fast aller Terminal-Emulationen mehr oder weniger identisch. Als User müssen Sie sich darüber keine großen Sorgen machen, verwenden Sie einfach das Terminal das bei ihrer Distribution dabei ist. Abbildung 16 zeigt eine Terminal-Emulation mit dem **top**-Kommando auf einem LINUX-System.

Trotzdem hier eine kurze Liste einiger der verbreitetsten Terminal-Emulationen auf LINUX:

```
(1)     konsole           (Default bei KDE)
(2)     xterm             (x11 Default)
(3)     gnome-terminal    (Default bei GNOME)
(4)     rxvt
(5)     dtterm            (Default bei CDE)
(6)     hpterm            (nur auf HP-UX)
(7)     aterm
```

```
Wenn es einmal passiert, ist
 es ein Bug. Wenn es zweimal
passiert, ist es ein Feature.
 Wenn es öffters passiert ist
  es eine Design Philosophie
```

15 Die Shell

Die Shell legt fest, in welcher Weise (Syntax) sich die Konsole programmieren lässt. Wie bereits erwähnt, bietet die Konsole einen programmierbaren Kommandointerpreter um Prozesse zu starten und zu steuern. Wie genau dieses Programmierinterface zur Konsole aussieht und funktioniert wird von der sogenannten Shell festgelegt. Es gibt natürlich mehr als eine Shell. Warum eine Shell wenn man auch fünf haben kann? Die gebräuchlichsten Shells unter LINUX sind:

```
(1) sh    /bin/sh    Bourne-Shell         klassische LINUX/UNIX-Shell
(2) bash  /bin/bash  Bourne again shell   Erweiterung der 'sh', sehr verbreitete Shell
(3) csh   /bin/csh   C-Shell              Shell die C-Syntax verwendet
(4) tcsh  /bin/csh   Tenex C-Shell        C-Shell kompatibel
(5) ksh   /bin/ksh   Korn-Shell           vorallem bei Administratoren beliebt ist
```

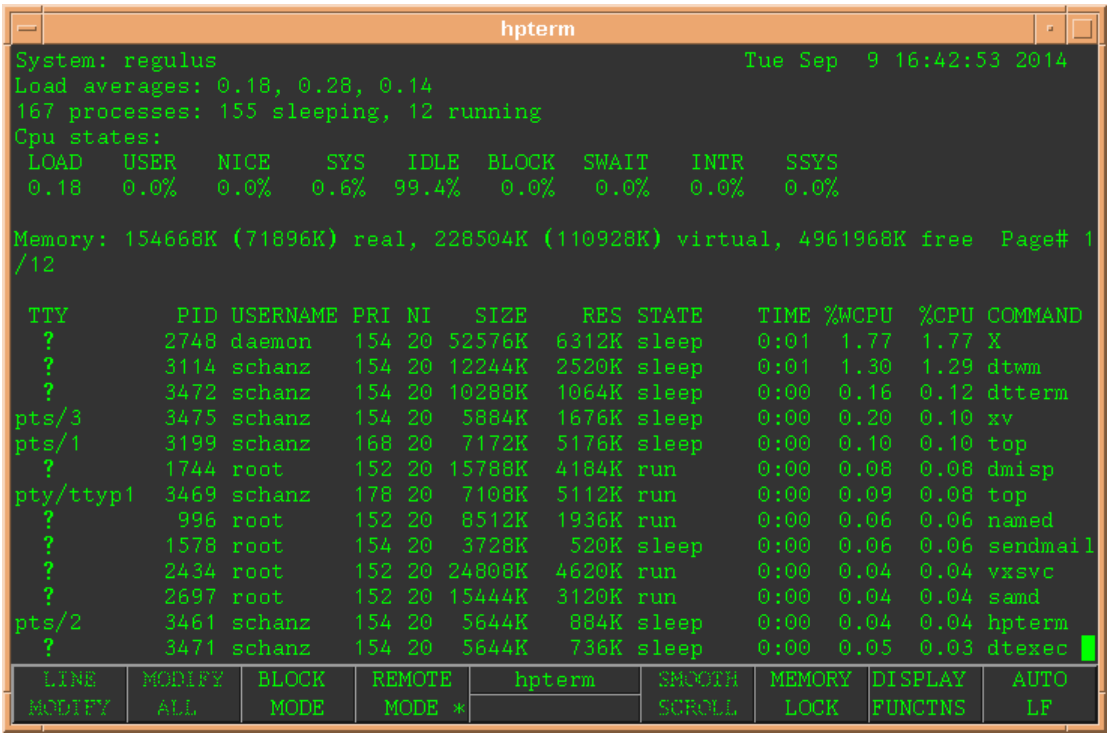

Abbildung 16: *Terminalemulation eines HP-Terminals unter HP-UX. Zu sehen ist die Ausgabe des 'top'-Kommandos auf der Konsole unter User 'schanz' auf dem UNIX-Server 'regulus'. Wenn Sie die hier abgebildete Terminalemulation mit Abbildung 15 vergleichen, erkennen Sie die Ähnlichkeit mit dem Hardware-Terminal.*

```
(6) zsh   /bin/zsh   Z-Shell              orientiert sich an der 'ksh', leistungsfaehig

(7) xonsh /bin/xonsh Python-Shell         Shell die Python-Syntax verwendet
```

Jede dieser Shells besitzt eigene Ressource-Dateien die als Punkt-Dateien im jeweiligen User-HOME-Verzeichnis angelegt werden, so z.B.: '.bashrc' für die 'bash' oder '.csh' für die 'csh'. In den Ressource-Dateien können Vereinbarungen zu Umgebungsvariablen wie z.B. $PATH und Kommando-Aliase eingetragen werden.

Ausschnitt aus der '.bashrc', der Ressource-Datei der 'bash'-Shell:

```
############ bashrc ##########

# Set up finder
      function findx ()
      { grep < /.log/findDB $* | more;
      }

# Set up pager
      PAGER=/bin/more
      export PAGER

# include dot in search path, but not for root !

# Set up the search paths:
```

```
PATH=$HOME/.dt/bin:/bin:/usr/bin:/sbin:/usr/sbin:/usr/bin/X11:/usr/local/bin:/us
r/local/bin/X11:/usr/contrib/bin:/usr/contrib/bin/X11:/usr/dt/bin:$PATH
export PATH

PATH=$HOME/.dt/bin:$PATH
export PATH

# Set up the Manpath:
MANPATH=/usr/man:/usr/local/man:/net/usr/man:$MANPATH
export MANPATH

# Set prompt:
if [ -O /etc/passwd ]; then
        export PS1="\$system:\[\W] # "
else
        export PS1="\h:[\W] > "
fi

# Aliase
alias ll='ls -l'
alias cls=clear
alias h=history
alias md=mkdir
alias rd=rmdir
alias dir=ls
alias copy=cp
alias m=more
alias del=rm
alias da='ls -al | more'
alias rename=mv
alias et=$HOME/.dt/bin/defedit

stty erase "^H" kill "^U" intr "^C" eof "^D" susp "^Z" hupcl ixon ixoff tostop

EDITOR=/usr/bin/vi
export EDITOR

BROWSER_PATH=/usr/local/bin/firefox
export BROWSER_PATH
```

Die meisten Shells legen eine Liste der letzten eingegebenen Kommandos als sogenannte 'history' an. Die Eingabe des Kommandos **'history'** (wenn Sie die 'bash' verwenden) gibt diese Liste aus. Man kann dort die Eingabe von früheren Kommandos nachschlagen. Der Aufruf der letzten Kommandos ist auch einfach durch drücken der Pfeil-Tasten (auf und ab) auf der Tastatur möglich. Die meisten Shells verwenden auch eine Befehlszeilenergänzung. Es reicht dann aus, den ersten Buchstaben des Befehls einzugeben und dann auf die 'tab'-Taste der Tastatur zu drücken. Die Shell ergänzt dann den Befehl oder macht Vorschläge. Die Funktion ist sehr hilfreich und beschleunigt die Kommandoeingabe beträchtlich! Der Inhalt der History sieht in etwa so aus:

```
 :    :
492  man man
493  uname -a
494  ps
495  man ps
496  su -
497  ls -al
498  ssh root@rigelD
499  grep root /etc/passwd
500  vi .bashrc
501  history
```

Viele Kommandos senden ihre Ausgabe nach „stdio“ (Standard I/O), d.h. direkt an die Terminal-Konsole. Das ist insbesondere bei längeren Ausgaben wie bei ’history’ problematisch, weil das Terminal durchscrollt.

Möchten Sie die Ausgabe eines Kommandos speichern, so können Sie die Ausgabe von 'stdio' auch mit dem '>'-Symbol in eine Datei umleiten, also z.B.:

```
[~]> history > meinehistorydatei.log
```

Dies leitet die gesamte Ausgabe von 'history' in die Datei 'meinehistorydatei.log' um, die zu diesem Zweck automatisch neu angelegt wird. So können Sie die Shell-History oder auch die Suche nach ihren jpg-Dateien zum Beispiel in ein Log-File umleiten, dass sie später in aller Ruhe mit einem Editor öffnen und weiter bearbeiten oder durchsuchen können. Oder Sie verwenden das ’grep’-Kommando um z.B. nach dem String ’jpg’ zu suchen:

```
[~]> grep jpg meinehistorydatei.log
```

Bei jedem neuen Aufruf einer solchen Ausgabe 'history > meinehistorydatei.log' wird die Datei 'meinehistorydatei.log' neu angelegt, die alte Datei geht dabei verloren. Möchten Sie, dass die Ausgabe von 'stdio' an eine bestehende Datei angehängt wird ohne den bereits vorhandenen Inhalt zu überschreiben, dann verwenden Sie statt '>' das Symbol '>>'.

```
[~]> history >> meinehistorydatei.log
```

Dies hängt die ’stdio’-Ausgabe des Kommandos einfach an das Ende der vorherigen Ausgabe, die sich bereits in ’meinehistorydatei.log’ befindet, an.

Jeder hinreichend komplexe Fehler ist ununterscheidbar von einem Feature. – Rich Kulawiec

16 Pipes

Pipes '|' verbinden die Ausgabe von Kommandos mit der Eingabe anderer Kommandos. Das ist eine clevere Art Shell-Skripte zu vermeiden. Natürlich können Sie auch ein Skript schreiben, dass mehrere Kommandos enthält und nacheinander ausführt. Was die Übergabe der Daten von einem Kommando zum anderen betrifft (zum Beispiel über eine temporäre Datei oder Variablen) müssen Sie sich im Fall von Skripten aber selbst kümmern. Mit Pipes ist die Sache einfacher, Pipes verbinden die Ausgabe von Kommandos mit der Eingabe von anderen Kommandos auf sehr elegante weise.

Angenommen Sie möchten wissen wie viele ihrer Bilder, die sich in Ihrem User-HOME-Verzeichnis befinden mögen, vom Typ 'jpg' sind. Mit dem Kommando 'ls' können Sie alle Dateien in Ihrem User-HOME-Verzeichnis anzeigen lassen. Gehen Sie also in ihr User-HOME-Verzeichnis:

```
[~]> cd
```

Wenn Sie das 'ls'-Kommando eingeben können Sie sich die Dateien in ihrem HOME-Verzeichnis anzeigen lassen, aber wahrscheinlich gibt es da gar keine Bilder im 'root'-Level ihres HOME-Verzeichnisses. Vielleicht gibt es aber 'jpg'-Dateien in einem der Unterverzeichnisse oder in einem der versteckten Verzeichnissen (die deren Name mit einem '.' anfängt). Sie könnten das 'find'-Kommando verwenden um nach solchen Dateien zu suchen:

```
[~]> find .
```

Dies gibt Ihnen eine lange Liste von Dateien aus, sie enthält alle Dateien in allen Unterverzeichnissen in Ihrem User-HOME-Verzeichnis. Um aus dieser Liste die 'jpg'-Dateien zu filtern können Sie das 'grep'-Kommando verwenden. Dazu verbinden Sie das 'grep'-Kommando über eine PIPE direkt mit der Ausgabe des 'find'-Kommandos:

```
[~]> find . | grep .jpg
```

Das 'grep'-Kommandos filtert alle Dateien aus der Liste die 'find' ausgibt, die den String '.jpg' enthalten. Wenn Sie wollen können Sie diese Liste alphabetisch sortieren unter Verwendung des 'sort'-Kommandos. Hierzu hängen Sie das 'sort'-Kommando mit einer weiteren PIPE and die Ausgabe des obigen Aufrufs:

```
[~]> find . | grep .jpg | sort
```

Das Ergebnis ist eine alphabetisch sortierte Liste aller 'jpg'-Dateien innerhalb Ihres HOME-Verzeichnisses mit allen Unterverzeichnissen. Wenn Sie wollen können Sie diese Liste für später in eine Datei umleiten:

```
[~]> find . | grep .jpg | sort > meine_sortierten_jpg_dateien.log
```

Möchten Sie wissen wie viele 'jpg'-Dateien Sie in dieser Liste haben, dann können Sie das 'wc' (word count) -Kommando verwendet und die Einträge in der Liste zählen lassen. Verbinden Sie hierzu die Ausgabe der Liste mit einer weiteren PIPE mit der Eingabe des **'wc**'-Kommandos:

```
[~]> find . | grep .jpg | sort | wc -l
```

Der Befehlsaufruf gibt Ihnen eine einzige Zahl zurück, die Anzahl der '.jpg'-Dateien in Ihrem User-HOME-Verzeichnis. Wenn Sie wissen wollen wieviele '.jpg'-Dateien sich auf dem gesamten Dateisystem befinden können Sie 'find .' durch 'find /' ersetzen und statt im HOME-Verzeichnis unterhalb des 'root'-Verzeichnisses '/' suchen lassen:

```
[~]> find / | grep .jpg | sort | wc -l
```

Wenn Sie dies versuchen sehen wahrscheinlich eine Menge Fehlermeldungen des Typs 'Permission Denied'. Das liegt daran, daß Sie als normaler User für viele Verzeichnisse des Dateisystems keine Berechtigung haben. Der Kommandoaufruf zählt also nur die Dateien in Verzeichnissen die offen sind. Ausserdem wird es wahrscheinlich eine Zeit lang dauern bis der Aufruf das gesamte Dateisystem durchsucht hat. Wenn ich das Kommando auf meinem Dateiserver ausführe erhalte ich nach ein paar Minuten die Zahl: 211096.

Sie können den PIPE Mechanismus auch verwenden wenn die Konsoleausgabe von manchen Kommandos zu lang ist um im Terminal vollständig angezeigt zu werden, wenn das Terminal bei der Ausgabe also durchscrollt. Verwenden Sie dann einfach einen Textbrowser wie 'more' oder 'less' und leiten Sie die Ausgabe des Kommandoaufrufs mit einer PIPE in den Textbrowser um, hier z.B. für das 'history'-Kommando der Shell:

```
[~]> history | less
```

Der Textbrowser gestattet es Ihnen mit den Cursortasten der Tastatur in der Ausgabe von 'history | less' zu blättern. Selbst wenn Sie nur das 'ls'-Kommando mit der Option '-al' starten wird die Liste vermutlich bereits zu lang für das Terminal und die Verwendung des Textbrowsers ist angezeigt:

```
[~]> ls -al | less
```

Zusammen mit dem PIPE Mechanismus bilden die Konsole-Kommandos von LINUX eine Art Baukasten mit dem Sie erstaunliche Dinge vollbringen können, einfach durch die Kombination der richtigen Kommandos in der richtigen Reihenfolge und der Verwendung der PIPE. Versuchen Sie es!

```
Ich glaube es gibt einen
Weltmarkt für ungefähr fünf
Computer. - attr. Thomas J.
Watson (Chairman of the
Board, IBM), 1943
```

17 Editore

Was ist ein Editor?

Ein Editor ist eine Software mit der Sie den Inhalt einer Datei verändern können. Die Art von Editor die Sie am meisten verwenden werden ist der Texteditor. Mit dem Texteditor können Sie 'lesbare' Dateien öffnen und bearbeiten. Bei LINUX gibt es 'lesbare' und 'nicht-lesbare'-Dateien. Die 'nicht-lesbaren'-Dateien sind meistens in Maschinencode (binär) und zur Ausführung bestimmt (Programme oder sie speichern Daten für Bilder, Filme, Sounds usw. oder z.B. den KERNEL selbst). Die Bezeichnung 'lesbare' und 'nicht-lesbare' bezieht sich hier auf die Verwendung mit Texteditoren.

Die häufigste Verwendung für 'lesbare' Dateien sind Konfigurationsdateien für Software oder für LINUX selbst, sowie Beschreibungen und Anleitungen. Diese können mit einem Texteditor geöffnet und bearbeitet werden. Der größte Teil der Konfiguration des LINUX-Betriebssystems wird über solche 'Config-Dateien' gesteuert.

Die globalen 'Config-Dateien' von LINUX liegen unter '/etc' des Dateisystems, die User-abhängigen lokalen Config-Dateien für Software liegen im jeweiligen User-HOME-Verzeichnis '/home/<username>/'. Wenn Sie keine SuperUser-Rechte auf dem System haben sind für Sie nur die lokalen 'Config-Dateien' von Interesse. Wenn Sie in ihrem

User-HOME-Verzeichnis sind (cd) und dort das Kommando ’ls -al | less’ eingeben, sehen Sie viele Dateien die mit einem ’.’ anfangen. Dabei handelt es sich größtenteils um ’Config-Dateien’ für Software. Sie können einen Texteditor verwenden um dieser Dateien zu bearbeiten.

Unter LINUX gebräuchliche Texteditore sind:

```
(1) 'vi'       : laeuft in der Konsole, gilt bei Einsteigern als kryptisch und schwer zu
                 bedienen, ist dafuer grandios kompakt und leistungsfaehig!

(2) 'emacs'    : laeuft in der Konsole oder in einem grafischen Fenster, ist un-
                 glaublich vielseitig und ermoeglicht Dinge die weit ueber die
                 Funktion eines normalen Editors hinaus gehen.

(3) 'nedit'    : grafischer Texteditor, kompakt, leicht zu verstehen

(4) 'gedit'    : grafischer Texteditor von GNOME, kompakt, leicht zu verstehen

(5) 'kate'     : grafischer Texteditor, sehr leistungsf"ahig, gro"s und langsam

(6) 'dtpad'    : grafischer Texteditor, nur auf CDE Installationen

(7) 'pico'     : einfacher Konsolen basierter Editor
```

Sie sollten einen Texteditor erlernen, der auch rein auf der Konsole funktioniert. Angenommen Sie wollen Dateien auf einem System editieren das keine Grafikausgabe unterstützt, wie zum Beispiel s.g. Embedded-Systeme, Router, Micro-Controller, der Raspberry-Pi oder Sie haben eine Secure-Shell-Verbindung (ssh) zu einem entfernten Computer, die keine Grafiknutzung erlaubt, oder Ihr System ist nach einem Absturz einfach defekt und erlaubt vorübergehend keine Grafikausgabe. In all diesen Fällen ist ein grafischer Texteditor nutzlos. Mit einem Konsole basierten Texteditor hingegen sind Sie selbst in solchen Situationen fähig Dateien zu editieren und können das System vielleicht reparieren. Abbildung 18 zeigt einen Screenshot mit vier geöffneten Texteditoren auf dem LINUX ’LXDE’-Desktop: ’emacs’, ’vi’, ’pico’ und ’kate’.

Wenn Sie eine Datei mit dem Texteditor öffnen wollen, z.B. die ’Config-Datei’ der Bourne-Shell ’.bashrc’, dann geben Sie einfach den Namen des Texteditors (hier z.B.: ’gedit’) zusammen mit dem Namen der Datei an:

```
[~]> gedit .bashrc
```

```
Ich weis endlich was 'abwärts
 Kompatibilität' bedeutet. Es
bedeutet, daß wir alle unsere
   alten Fehler behalten. -
      Dennie van Tassel
```

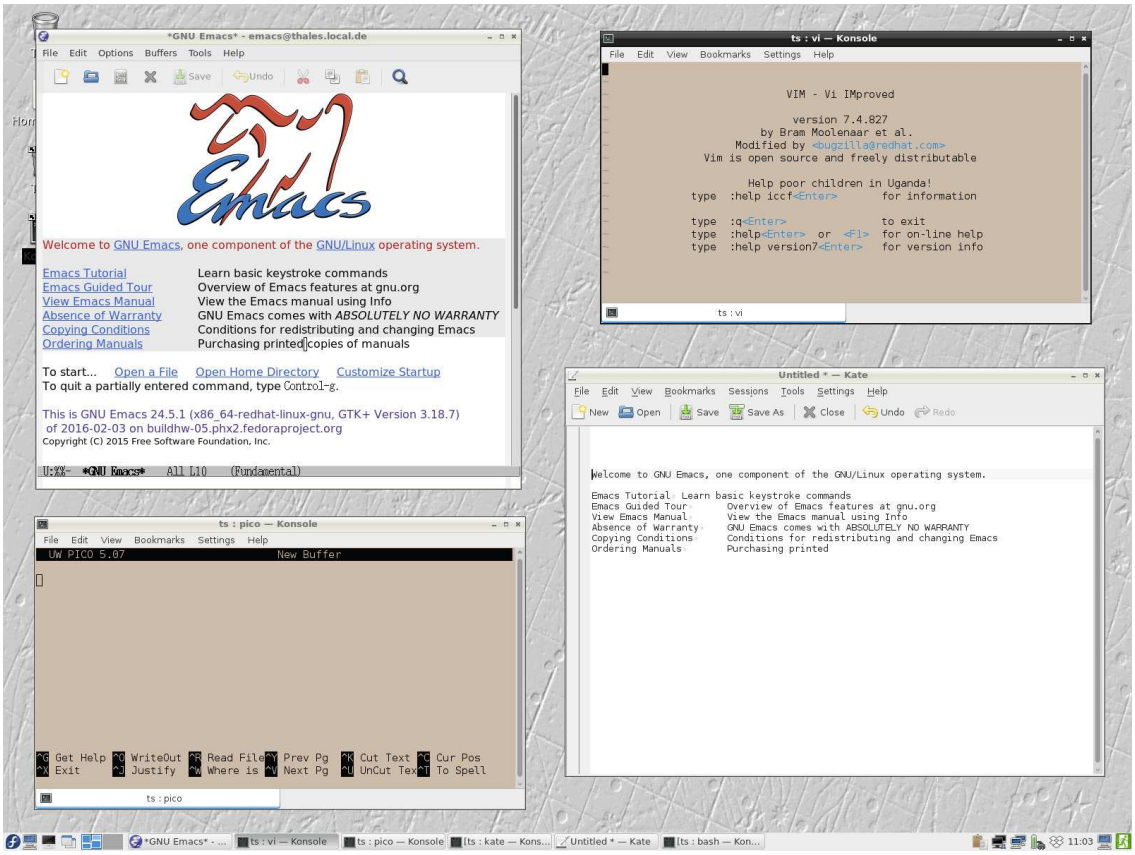

Abbildung 17: *Screenshot eines LINUX-Systems mit Editoren. Oben links: 'emacs', oben rechts: vi', unten links: 'pico', unten rechts: 'kate'. Das 'copy' und 'paste' über die Zwischenablage funktioniert einfach in dem das betreffenden Textfeld mit der linken Maustaste markiert (hier in 'emacs' -> grauer Bereich) und dann mit der mittleren Maustaste in der Zielanwendung (hier in 'kate') abgelegt wird.*

18 Die Zwischenablage

Die Zwischenablage ('Clipboard') erlaubt Ihnen das einfache kopieren von Textfeldern von einer Textanwendung in eine andere, also z.B. von einem Webbrowser in einen Texteditor oder in ein Textverarbeitungprogramm. Unter LINUX funktioniert die Zwischenablage jedoch anders als Sie das vielleicht von WINDOWS gewohnt sind. Bei LINUX-Systemen wird die Zwischenablage rein mit den Maustasten betrieben, die Tastenkombination 'ctrl c', 'ctrl v' hat bei LINUX eine andere Funktion. Zum 'copy' wird die linke Maustaste verwendet, mit der die betreffenden Textfelder markiert und in die Zwischenablage kopiert werden. Zum 'paste' wird über dem Zielfenster die mittlere Maustaste gedrückt. Der zuvor markierte Textbereich wird dann aus der Zwischenablage in das Zielfenster an die Stelle des Cursors kopiert. Manche Programme wie OpenOffice, die aus der WINDOWS-Welt stammen, verwenden aber auch unter LINUX den 'ctrl c' 'ctrl v' Mechanismus von WINDOWS.

Die Tastenkombination 'ctrl c' hat in der Konsole von LINUX eine andere Bedeutung, 'ctrl c' kopiert nicht, sondern beendet den aktuellen Kind-Prozess auf der Konsole! Wenn

Sie z.B. den Texteditor ’emacs’ auf der Konsole starten:

```
[~]> emacs
```

anschließend in die Konsole klicken und auf der Tastatur die Tastenkombination 'ctrl c' drücken, so wird der Kind-Prozess ’emacs’ sofort beendet! Wenn Sie anschließend noch 'ctrl d' drücken wird auch noch die Konsole geschlossen.

19 Programme

Wie Software installiert wird ist von der LINUX-Distribution abhängig. Debian-basierte Systeme verwenden den 'apt' Paket-Manager, Fedora-basierte Systeme verwenden 'yum'. Einen guten Überblick über die Software-Situation (installierte und installierbare Software) bieten grafische Installer wie 'synaptic' für Debian-Systeme oder 'yumex-dnf' für Fedora-Systeme. Hier kann die vorhandene bereits installierte Software, sowie Software aus externen Repositories im Internet per Mausklick ausgewählt und bei Bedarf entfernt oder hinzu installiert werden. Die Installation eines solchen grafischen Installers ist sehr sinnvoll und empfehlenswert! Gerade für Anfänger wird die zusätzliche Installation neuer Software damit erheblich erleichtert.

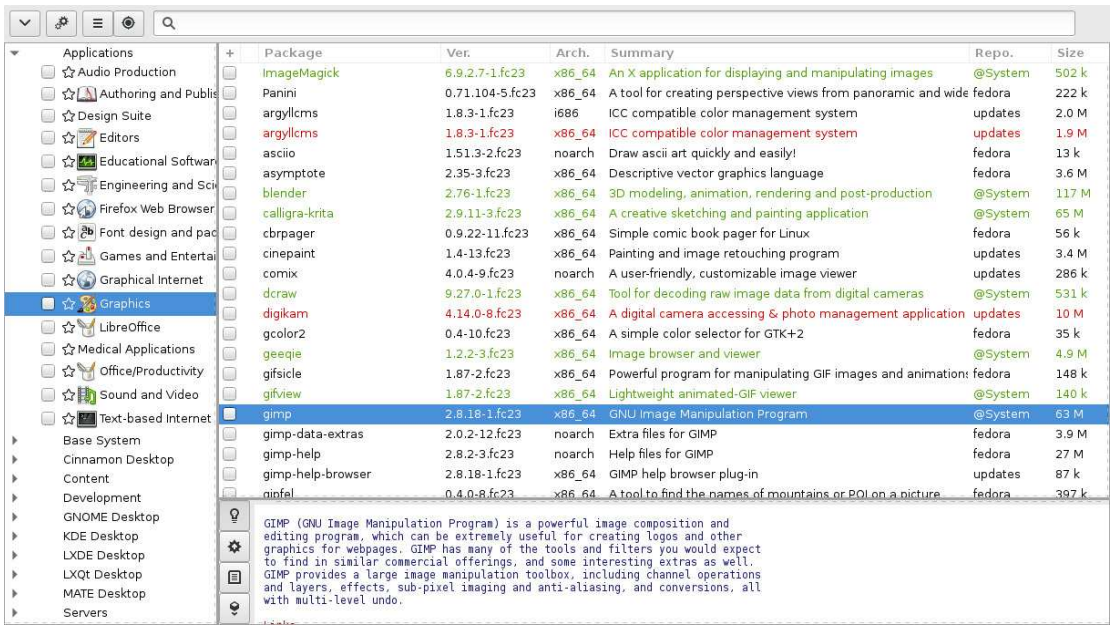

Abbildung 18: *YUMEX bietet eine einfache grafische Oberfläche zur Softwaresuche und zur Softwareinstallation auf Fedora (rpm) basierten LINUX-Systemen.*

Sollte sich 'synaptic' nicht auf Ihrem Debian-basierten System befinden, versuchen Sie die Konsoleneingabe: 'apt-get install synaptic' bzw. 'sudo apt-get install synaptic'

Sollte sich 'yumex-dnf' nicht auf Ihrem Fedora-basierten System befinden, versuchen Sie die Konsoleneingabe: ' dnf install yumex-dnf' bzw. 'sudo dnf install yumex-dnf'

ACHTUNG!

Die Installation von Software betrifft das gesamte System (also auch andere User) und erfordert deshalb in der Regel SuperUser Privilegien! User können an der Software-Ausstattung von LINUX-Systemen nichts ändern. Sollten Sie eine spezielle Software oder das Update einer Software benötigen → fragen Sie ihren Administrator!

In begrenztem Umfang besteht die Möglichkeit Software lokal ins eigenen User-HOME-Verzeichnis zu installieren. Von umfangreicheren Installationen ist jedoch aus Speicherplatzgründen (Quotas) abzuraten. Diese sollten vom SuperUser für das gesamte System installiert werden.

```
würden Häuser auf die gleiche
weise gebaut, wie Software
geschrieben wird, der erste
Specht der vorbei kommt, und
die Zivilisation liegt in
trümmern.
```

20 Administration

Die Administration von LINUX-Systemen stößt an die Grenzen dieser Anleitung. Leider gibt es bis heute kein einheitliches Administrationstool unter LINUX, jede Distribution bastelt hier ihre eigenen Lösungen oder verlässt sich gleich auf die manuelle Administration des Systems via Konsole.

Unter Administration fallen z.B. Aufgaben wie:

(1) User anlegen oder löschen
(2) Festplatten partitionieren, formatieren, Dateisysteme montieren, LVM usw.
(3) Netzwerke konfigurieren
(4) Backup
(5) neue Hardware einbinden
(6) Sprach und Keyboardeinstellungen ändern
(7) Uhrzeit und Datum ändern
(8) Firewalleinstellungen konfigurieren
(9) Drucker und Drucker-Queues einrichten
(10) Bildschirmauflösung ändern
(11) Dienste konfigurieren (cron/automounter/cups/nfs/samba/sshd usw ...)
(12) Daemons konfigurieren, Server konfigurieren ...
(13) Installation und Upgrade von Software...
usw....

Grundsätzlich geschieht die Administration größtenteils manuell über das Editieren und die Anpassung der Config-Dateien der verschiedenen Dienste und Applikationen von LINUX im Verzeichnis „/etc“ des Dateisystems. In der Regel erfordern diese Aufgaben aber Expertenwissen! Ich empfehle und verweise auf die entsprechenden Foren im Internet! Sie sollten aber unbedingt darauf achten, dass Sie bei der Suche nur solche Anleitungen zu der von Ihnen installierten Distribution zu rate ziehen, LINUX kann von Distribution zu Distribution sehr unterschiedlich sein!

Auf Fedora basierten Systemen gibt es zum Teil zahlreiche grafische Admin-Tools für diverse Aufgaben. Diese Tools beginnen alle mit dem Wort 'system-...'. Geben Sie in die Konsole das Wort 'system' ein und drücken Sie die 'tab'-Taste auf der Tastatur. Wenn Sie die 'bash' verwenden erhalten Sie eine Reihe von Vorschlägen wie z.B.:

```
system-config-boot
system-config-users
system-config-firewall
system-config-network
system-config-services
...
```

Mit diesen Tools können auch Anfänger große Teile des LINUX-Systems problemlos administrieren.

Das mit Abstand vollständigste und beste Admin-Tool unter LINUX-Systemen besitzt die SUSE-Distribution.'Yet Another Setup Tool' oder kurz 'yast' bietet ein vollständig Maus gesteuertes grafisches Tool für alle Belange der System-Administration des LINUX-Betriebssystems, siehe auch Abbildung 19. Ein so vorbildliches Tool findet man sonst nur unter kommerziellen UNIX-Betriebssystemen (z.B. 'smit' auf AIX oder 'sam' auf HP-UX).

Für alle anderen Distributionen empfehle ich die Admin- bzw. Config-Menüs, die Desktop-Umgebungen wie KDE oder GNOME mitbringen zu verwenden. Diese Tools bieten zumindest in Teilen grafische Menüs zur Administration vieler Aufgaben.

```
es gibt nie einen Bug,
 solange Sie ihn nicht
    gefunden haben
```

21 Installation

Zum Schluss noch ein Hinweis zur Installation. Sollten Sie LINUX selbst installieren, dann achten Sie darauf, dass Sie das Development-Paket ihrer Distribution mit installieren. Sollten Sie später jemals in den Bedarf kommen eine Software selbst zu compilieren, oder eine KERNEL-Erweiterung benötigen, so ist ein installiertes Development-Paket erforderlich. Wenn Sie z.B. einen Grafiktreiber für Ihre Grafikkarte vom Hersteller Herunterladen, z.B. weil Sie OpenGL-Unterstützung benötigen, so kann es nötig sein, dass

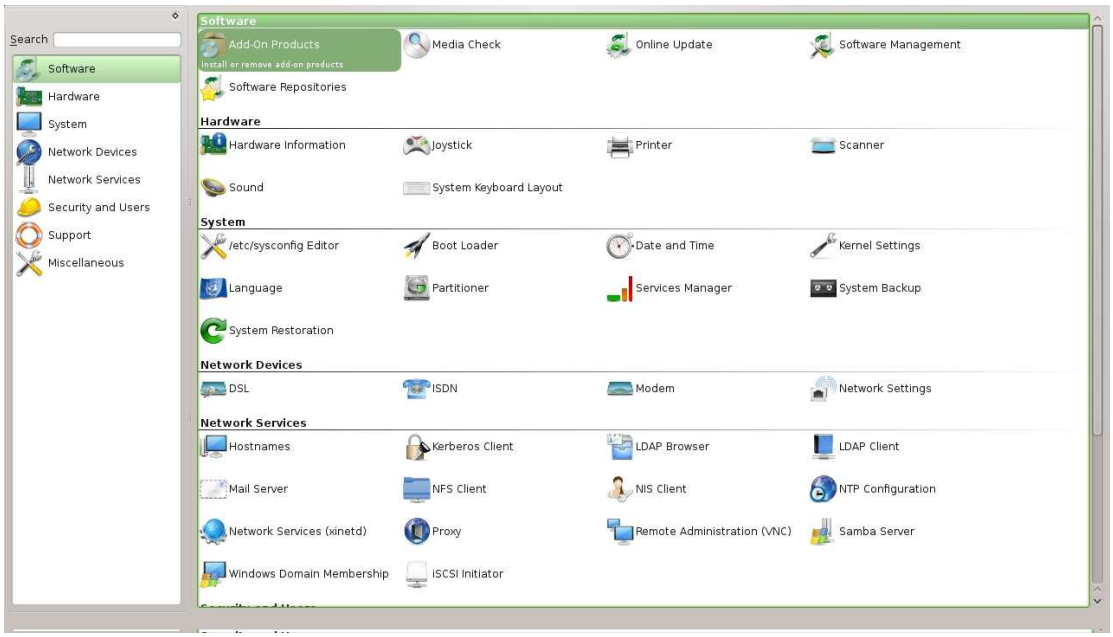

Abbildung 19: *Screenshot des YAST-Systemadministrations Managers von SUSE-LINUX (OpenSUSE)*

KERNEL-Module nachträglich kompiliert werden müssen. Dies geschieht bei der Installation des Treibers zum Teil automatisch, ist aber nur möglich wenn ein Compiler und die notwendige Software-Entwicklungsumgebung auf Ihrem System bereits installiert sind. Sie können testen ob ein Entwicklungssystem installiert ist. Geben Sie hierzu in der Konsole ein:

```
[~]> gcc -v
```

Sollten Sie darauf eine Ausgabe erhalten, die nicht 'command not found' lautet, so ist ein Entwicklungssystem höchst wahrscheinlich bereits installiert.

Sie sollten außerdem prüfen, ob Sie die richtigen Grafiktreiber für OpenGL-Suport installiert haben. Viele neue Softwarepakete erfordern OpenGL Unterstützung. Sollten Programme wie 'GOOGLE-EARTH' oder 'STELLARIUM' in der Grafikausgabe 'rukkeln', dann haben Sie wahrscheinlich nicht die richtigen Grafiktreiber. Vergewissern Sie sich im Repository ihrer LINUX-Distribution, ob OpenGL-Treiber für Ihr System und ihre Grafikhardware installiert werden können.

Wenn Sie LINUX zu Hause testhalber ausprobieren wollen, dann machen Sie das auf einem Rechner den sie nicht für Ihre Arbeit benötigen und auf dem keine Ihrer wertvollen Daten liegen. Am Anfang passieren Unfälle, Sie werden den Rechner vielleicht mehrfach neu installieren müssen! LINUX benötigt nicht so viele Ressourcen wie kommerzielle PC-Betriebssysteme und läuft auch auf älterer Hardware noch flüssig. Verwenden Sie

einen alten Rechner, den Sie als Versuchslabor verwenden und mit dem Sie das LINUX-Universum in Ruhe erforschen können!

Dieses Dokument wurde vollständig mit LINUX erstellt (Fedora 23)! Die hierbei verwendete Software war:

```
(1) der Editor 'vi'
(2) 'texlive' LaTeX Textbeschreibungssprache
(3) 'xlatex' LaTeX Bedienkonsole
(4) 'xfig' Vektorzeichenprogramm
(5) 'gimp' Bildbearbeitungssoftware
(6) 'xv' Bildbearbeitungssoftware
(7) 'convert' Bild-Konvertersoftware
(8) 'ispell' Spellchecker
(9) 'firefox' Internet Web-Browser
```

Das Programm ist erst dann
fehlerfrei, wenn der letzte
User tot ist