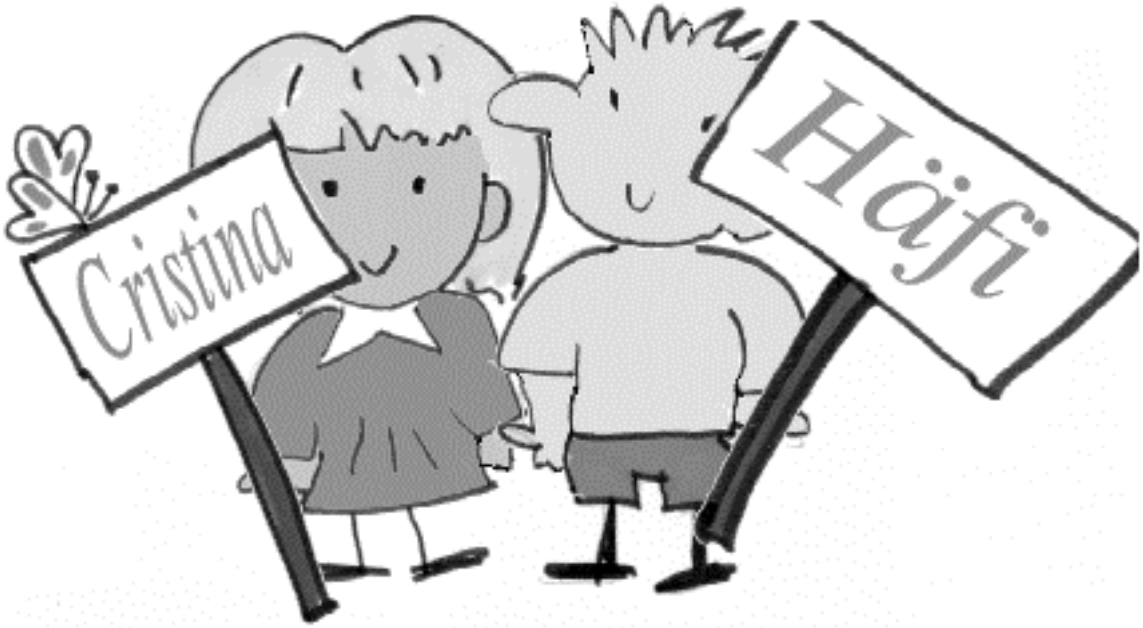
# *Die Abenteuer von...*




## *Einführung in UNIX*

Ein Leitprogramm in Informatik

**ETH-Leitprogramm "Einführung in UNIX"**
Version: September 1998

**Stufe, Schulbereich**
Universitäten, Fachhochschulen, individuelle Fortbildung

**Fachliche Vorkenntnisse**
PC-Anwender-Grundkenntnisse

**Bearbeitungsdauer**
ca. 5 Stunden

**Hilfsmittel**
Nebst theoretischen Teilen ist dieses Leitprogramm sehr auf die Praxis ausgerichtet. Man sollte daher ein Benutzerkonto auf einem UNIX-System zur Verfügung haben.

**Empfehlung für die Lehrperson**
Den Benutzern ist ein System zur Verfügung zu stellen, dass es ihnen ermöglicht, dieses Leitprogramm in der vorliegenden Form durchzuarbeiten. Andernfalls muss explizit auf die Besonderheiten des betreffenden Systems hingewiesen werden.

**Autoren**
Cristina Besomi, Christian Häfeli

**Herausgeber**
Werner Hartmann, Theoretische Informatik, ETH-Zentrum, 8092 Zürich

Die *ETH-Leitprogramme* in Informatik sind ein Gemeinschaftsprojekt von Karl Frey und Angela Frey-Eiling (Initiatoren), Werner Hartmann (Informatik), zusammen mit den Autorinnen und Autoren.

# Inhaltsverzeichnis

# Arbeitsanleitung

Das vorliegende Leitprogramm ist für das selbständige Durcharbeiten gedacht. Es führt in fünf Kapiteln in die Geheimnisse des Betriebssystems UNIX ein. In einem Additum können zusätzliche Kenntnisse erworben werden. Die Kapitel haben alle eine ähnliche Struktur:

**Lernziele** Was kann ich nach dem Bearbeiten dieses Kapitels?

**Theorie** Theoretischer Teil eines Kapitels.

**Praxis** Praktische Arbeiten am Computer.

**Aufgaben** Habe ich alles verstanden?

# Kapitel 1: Was ist UNIX?

## Übersicht

Vielleicht hast du auch schon mit einem Betriebssystem gearbeitet, ohne dass du dir dessen bewusst gewesen bist. Vermutlich hast du schon mal etwas von *DOS* oder *Windows* gehört. Dies sind zwei berühmte Vertreter von Betriebssystemen. Im universitären Bereich trifft man zudem häufig auf UNIX.
Du wirst in diesem Kapitel erfahren, was ein Betriebssystem genau ist. Du wirst sehen, warum man ein solches braucht und weshalb Betriebssysteme auf allen Computern anzutreffen sind. Nachdem du einige Grundsätze von Betriebssystemen kennengelernt hast, werden wir uns UNIX zuwenden. Wo immer es verschiedene Produkte auf dem Markt gibt, haben alle ihre Vor- und Nachteile. Wie es damit bei UNIX steht, wirst du im zweiten Teil erfahren.

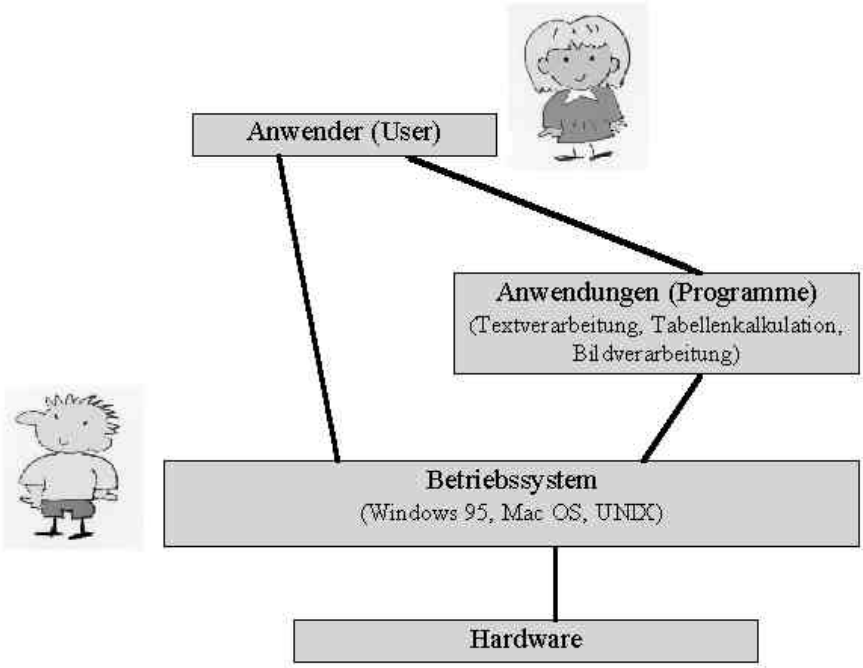
### Lernziele

Nachdem du dieses erste Kapitel durchgearbeitet hast, bist du in der Lage:

- Die Begriffe Hardware und Software zu erklären.
- Drei Aufgaben aufzuzählen, um die sich ein Betriebssystem zu kümmern hat.
- Zwei Besonderheiten von UNIX zu nennen und einem Laien zu erklären.

### Theorie

Zur Orientierung im Begriffsdschungel benutzen wir ein Schichtenmodell. Zuoberst steht die Anwenderin. In diesem Leitprogramm soll sie Cristina heissen. Zusammen mit Häfi wird sie dich durch dieses Leitprogramm begleiten. Häfi besitzt alle Fähigkeiten von UNIX. Er verkörpert das Betriebssystem, versehen mit einigen menschlichen Zügen. Er ist ein treuer Weggefährte von Cristina. Du wirst die beiden später noch besser kennenlernen.




**Hardware** Ein Rechner besteht typischerweise aus verschiedenen Komponenten. Alle Teile die materiell existieren (die man sehen oder anfassen kann) zählt man zur Hardware. Dazu gehören so bekannte Dinge wie Tastatur, Bildschirm, Prozessor und vieles mehr.

**Anwendung/Programm** Ein Programm ist ein Stück Text, auch Code genannt, welches ein Rechner verstehen kann. Wird es gestartet, so werden die darin enthaltenen Anweisungen ausgeführt. Ein ”laufendes” Programm nennt man auch **Prozess** (task, job). Dies darum, weil der Prozessor die eigentliche Arbeit bewältigt. Programme sagen also im Prinzip der Hardware was sie zu tun hat. Sie sind eine Anleitung, ähnlich einem Kochrezept.

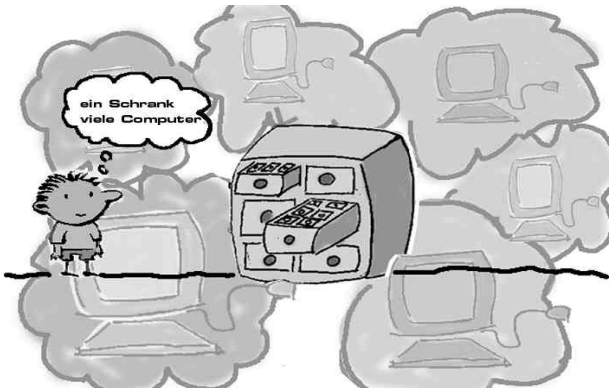
**Software** Alleine mit der Hardware kann der Computer noch nicht arbeiten! Zum Empfangen und Verarbeiten unserer Wünsche braucht er zusätzliche Hilfe. Dafür sorgen Programme. Das Sammelsurium aller beteiligten Programme eines Systems nennt man Software.

**Betriebssystem** Damit der Computer (besser die Hardware) überhaupt einmal etwas tut, braucht er eine Art Basissoftware, welche einen Grundbetrieb garantiert. Ein Programm mit dieser Aufgabe nennt man Betriebssystem (Operating System, OS). Als unverzichtbares Stück Software stellt es eine Reihe von Grundfunktionen zur Verfügung. Unter anderem sorgt es für:

1. **Verwaltung der Hardware** Das Betriebssystem sorgt für ein organisiertes Zusammenspiel der verschiedenen Komponenten. Ein wichtiger Teil davon ist z.B. die Speicherung von Daten. Das Betriebssystem muss dafür sorgen, dass die Daten auf einem Speichermedium jederzeit abrufbar sind und dass man möglichst schnell auf sie zugreifen kann. Eine weitere wichtige Aufgabe ist das Ausgeben von Daten auf dem Bildschirm oder dem Drucker.
2. **Kontrolle von Programmen** Das Betriebssystem sorgt für die Ausführung der Programme und dafür, dass sich verschiedene Programme nicht in die Quere kommen.
3. **Befehlsverarbeitung** Dies ist die eigentliche Schnittstelle (der Kontakt) zum Benutzer. Via Tastatur oder Mausklick gibt der Benutzer seine Befehle an das Betriebsystem weiter. Hinter diesen Befehlen wiederum stecken meist kleinere Programme.

**Graphische Benutzeroberfläche – GUI (Graphical User Interface):**
Heute bedient man die meisten PC's (Personal Computer) über eine graphische Oberfläche. Früher wurden die Befehle per Tastatur eingegeben. Im Gegensatz dazu findet man heute oft eine Menge Fenster und Icons auf dem Bildschirm. Mit Mausklicken navigiert man zwischen den verschiedenen Fenster und Programmen hin und her. In der Entstehungsgeschichte von graphischen Oberflächen (auch Desktops genannt) erkennt man eindeutig den Trend Büroarbeitsplätze nachzuempfinden. Die Arbeitsfläche beinhaltet Ordner, Dokumente oder Abfalleimer. Der wohl berühmteste Verteter ist das Betriebssystem *Windows*, bei welchem die graphische Oberfläche ins System integriert ist.

**Netzwerk:**
Werden mehrere Computer miteinander verbunden, so bilden sie ein Netzwerk. Die Vorteile einer solchen Anordnung liegen auf der Hand. Daten müssen nicht mehr per Diskette von einem Gerät zum anderen transportiert werden. Die Computer können direkt miteinander kommunizieren. Es ist sogar möglich, eine Arbeit auf mehrere Computer zu verteilen. Ebenso werden Benutzerdaten oft nicht auf den einzelnen Maschinen gespeichert, sondern auf Festplatten, auf welche die Computer des Netzwerkes Zugriff haben. Diese Datenablage kann man mit einem grossen Schrank vergleichen. So sieht's auf jeden Fall Häfi!

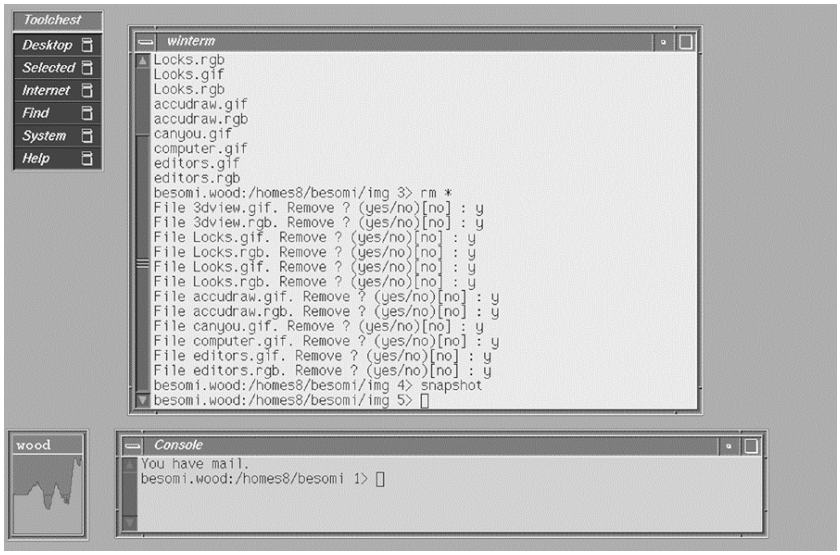



# Was ist UNIX?

UNIX ist eine grosse Familie von mehr oder weniger gleichartigen Betriebssystemen (BSD, HP-UX, SUN-OS, Linux, Solaris, etc). Wir werden hier nicht weiter auf die unterschiedlichen Richtungen eingehen. In der Folge werden wir von dem Betriebssystem UNIX sprechen, obwohl dies eigentlich nicht ganz korrekt ist. Dieses Vorgehen sei uns erlaubt, da man als UNIX Anfänger glücklicherweise von diesen Unterschieden nicht allzu viel merkt.

UNIX ist ein verhältnismäßig altes Betriebssystem. Die Geschichte von UNIX begann im Jahr 1969. Ken Thompson, ein Programmierer bei Bell Laboratories, begann das Betriebssystem MULTICS umzuschreiben. Er wollte das bestehende System, welches für Grossrechner gedacht war, für Kleinrechner umbauen und anpassen. Der eigentliche Name wurde dann 1970 von Brain Kernighan geprägt.

UNIX ist im Grunde ein befehlsorientiertes Betriebssystem. Man kann es durch einfache Textfenster ohne graphische Oberfläche bedienen. Verschiedene Hersteller haben in den letzten Jahren jedoch auch graphische Oberflächen für UNIX entwickelt.




**Wichtige Merkmale von UNIX**
Einer der Vorteile von UNIX ist die Fähigkeit verschiedene Prozesse gleichzeitig laufen zu lassen. Während im Hintergrund etwas berechnet wird, kann man bereits weitere Befehle eingeben und somit andere Programme starten. Die Möglichkeit mehrere Programme gleichzeitig ausführen zu können nennt man **Multitasking.**

Ein anderes auffallendes Merkmal: UNIX erlaubt es, dass an einem Rechner mehrere Leute gleichzeitig arbeiten können. Stell dir nun dabei nicht vor, dass mehrere Personen vor dem gleichen Bildschirm sitzen! Denke an folgenden Satz zurück: Bilden mehrere Computer ein Netzwerk, so hat ein Benutzer die Möglichkeit auf andere Rechner zuzugreifen. Konkret benutzt er dabei eigentlich die Rechenleistung eines anderen Computers, während er die Ausgaben weiterhin auf seinem Monitor sieht. UNIX stört es nicht, wenn mehrere Personen den gleichen Rechner belasten. Es sorgt dafür, dass der Betieb trotzdem einwandfrei funktioniert. Man sagt darum dieser Art des Rechnerbetriebs **Mehrbenutzer- oder Multiuser-Betrieb.**

# Aufgaben zur Lernkontrolle

In diesem ersten Kapitel hast du etwas über die wichtigsten Grundlagen der Informatik gelernt. Vielleicht war dies für dich alles ein alter Zopf! Falls nicht, so empfiehlt es sich kurz bei den folgenden Aufgaben zu verweilen. Sie werden dir das praktische Arbeiten vereinfachen.

### Aufgabe 1

Ein Kollege hat sich eben einen neuen Computer gekauft. Er zählt dir stolz auf, was er nun so alles zu seinem Besitz zählen darf. Dies sind eine Maus, Tetris (Computerspiel), ein Textverarbeitungsprogramm, ein Drucker und Windows 95. Er erzählt Dir weiter, dass er sowieso nur Software braucht. Er arbeite viel lieber nur mit Software. Was meinst du dazu? Wie stellst du dich zu dieser Aussage? Drei bis vier Sätze genügen als Antwort.

### Aufgabe 2

In einer Zeitung steht folgende Aussage: ”Beim Arbeiten an einem Computer braucht man immer Hard- und Software!”
Versuche einem Nicht-Informatiker zu erklären, warum diese Aussage richtig ist. Kleiner Tip: Denke an den Begriff ”Betriebssystem”.

### Aufgabe 3

Du weißt jetzt, was ein Betriebssystem ist. Du kennst die Grundaufgaben, die es zu erledigen hat. Finde ein entsprechendes Analogon dazu. Hier ein kleines Beispiel, damit du verstehst wie die Frage gemeint ist.
Franz behauptet: ”Eine Postleitstelle funktioniert eigentlich wie ein Betriebssystem. Die Briefkästen sowie die Poststellen und Fahrzeuge mit ihren Angestellten und Chauffeuren entsprechen der Hardware eines Computers. Die Leitstelle koordiniert deren Einsatz. Das Ziel ist, dass der Kunde, also der Benutzer möglichst zufrieden ist.

# Lösungen

### Aufgabe 1

Zur Hardware gehören: Maus, Drucker
Zur Software gehören: Betriebssystem, Word 6.0, Tetris, Windows 95
Er braucht zum Arbeiten sicher Hard- wie Software. Nur mit beidem zusammen lässt sich etwas anfangen.

### Aufgabe 2

Um überhaupt mit dem Rechner kommunizieren zu können, muss dieser mit einer Anzahl Grundprogramme ausgerüstet sein. Man fasst diese auch unter dem Namen Betriebssystem zusammen. Programme gehören zur Software. Sie instruieren die Hardware, was sie zu tun hat. Erst das Zusammenspiel von Hard- und Software stellt also dem Benutzer ein brauchbares System zur Verfügung.

### Aufgabe 3

Ein Beispiel aus der Welt des Sports:
Eine Tennisspielerin vertrete die Anwederseite. Ihr steht eine Infrastruktur (Hardware) zur Verfügung. Dazu gehört ein Platz, ein Schläger und Bälle. Das Spiel selber ist wie ein Programm. Zusammen mit den Regeln sorgt ein Schiedsrichter für einen reibungslosen Ablauf. Dies ist auch die Aufgabe eines Betriebssystems.

# Kapitel 2: login, das erste Anmelden!

## Übersicht

Du möchtest nun sicher einmal etwas am Computer tun. In diesem Kapitel lernst du die ersten Schritte, die man in einem UNIX-System befolgen muss. Nachdem du den Theorieteil durchgelesen hast, wirst du die ersten Erfahrungen in der Praxis direkt am Computer machen. Versuche die beschriebenen Schritte gleich 1:1 nachzuvollziehen.

### Lernziele

Nach diesem Kapitel wirst du über folgendes Bescheid wissen:

- Du weißt, warum jeder UNIX - Benutzer ein Konto und ein Passwort braucht.
- Du bist in der Lage, dich am System an- und abzumelden.

### Theorie

UNIX wird typischerweise in einer Netzwerkumgebung benutzt. Das heisst für dich, dass es keine Rolle spielt, an welchem Computer du arbeitest. Damit du aber immer deine Arbeitsumgebung mit deinen Daten vorfindest, muss der Computer wissen wer du bist! Darum braucht es eine eindeutige Identifikation.

**Systemadministrator** Diese Person ist der Chef über das Netzwerk, an dem du arbeitest. Er regelt den Zugriff auf das von ihm betreute Netzwerk. Von ihm erhälst du ein **Konto** (Account)**.** Er stellt dir einen vorgegeben festen Speicherplatz zur Verfügung, wo du deine Daten ablegen und verwalten kannst. Die ganze Registrierung ist absolut notwendig. Ohne einen Account kannst du nichts tun, denn UNIX lässt nur eingetragene Benutzer zum Arbeiten zu. Falls du jemals ein Problem hast, bei dem dir sonst niemand helfen kann, dann wende dich an den Systemadministrator!

**Anmelden (login)** Wenn du am System arbeiten willst, musst du dich identifizieren können. Das heisst, dass du dich als berechtigter Benutzer ausweisen musst. Stell dir vor, du stehst vor einem Bankautomaten. Dort brauchst du einen PIN, damit auch wirklich nur du von deinem Konto Geld abheben kannst. Ähnlich funktioniert dies auch unter UNIX. Du hättest wohl keine Freude, wenn jemand in deinen Daten herumwühlen würde!
Deshalb muss man einen **Login-Namen** und ein **Passwort** eingeben, bevor das Betriebssystem aufgestartet wird. Der Login-Name besteht meist aus Teilen deines richtigen Namens, während das Passwort eine beliebige Folge von Zeichen ist. Login und Passwort erhälst du ebenfalls vom Systemadministrator.

# Die Praxis

## Das erste Anmelden

**Grundsatz** Schalte nie einen Rechner aus! Eine Maschine wird üblicherweise nur vom Systemadministator nach einem vordefinierten Verfahren ausgeschaltet. Geschieht dies nicht auf diese Weise, so kann durch ein Ausschalten grosser Schaden angerichtet werden.

1. Setze dich an einen freien Arbeitsplatz.

2. Ist der Bildschirm schwarz, so läuft sehr wahrscheinlich ein Bildschirmschoner, der automatisch aktiv wird, wenn über längere Zeit keine Eingabe erfolgte. Drücke nun eine Taste, am besten SHIFT oder CTRL, da diese allein keine Eingabe verursachen. Der Tastendruck aktiviert nun den Bildschirm wieder. Hat dies nichts genützt, so ist der Bildschirm ausgeschaltet. Schalten ihn wieder ein. Bist du nicht ganz sicher, welches nun der Bildschirmschalter ist, dann frage jemanden. Ja nicht den Rechner ausschalten!

3. Normalerweise solltest du nun am Bilschirm etwa folgendes sehen:


Ist dies nicht der Fall, so frage jemanden. Es besteht noch die Möglichkeit, dass jemand die Arbeitsstation gesperrt hat, weil er nur schnell einige Minuten weg ist. Gib bei *Login:* deinen

Login-Namen und bei *Password:* dein Passwort ein. Ist ja logisch. Wenn du wirklich ein eingetragener User bist und dich nicht vertippt hast, dann wird nun deine Arbeitsumgebung aufgestartet. Zuerst siehst du oft noch die Mitteilungen des Tages. Dies sind kurze Mitteilungen zum System. Es lohnt sich diese jeweils schnell zu lesen. Danach wird wohl wie heute üblich eine graphische Oberfläche geöffnet.

**Übungsaufgabe**
Befindest du auf einer graphischen Oberfläche, so müsste irgendwo ein EXIT sein. Klickst du mit der Maus darauf, so steigst du wieder aus dem System aus. Hast du keine graphische Oberfläche, so musst du *logout* oder *exit* eintippen, um dich beim System abzumelden. Versuche dich nun einmal vom System abzumelden und danach nochmals anzumelden.

*Hoffentlich hat es geklappt! Falls du Probleme hast, dann frage jemanden in deiner Nähe. Sei unbesorgt, es gibt keinen UNIX Benutzer der sich im Lauf seiner Karriere nicht schon durchfragen musste. Dir wird es bestimmt nicht anders ergehen. Du kannst dir eine Menge Zeit und Ärger ersparen, wenn du dich nicht ums Fragen anderer Leute herumdrückst. Auf jeden Fall solltest du immer, wenn du im Verlauf dieses Leitprogramms nicht weiter kommst, diesen Weg einschlagen. Es ist gut möglich, dass nicht alles bis ins Detail,m it dem hier beschriebenen überein-stimmt. Hast du es aber einmal herausgefunden, so wirst du es nicht so schnell wieder vergessen. Alle waren einmal Anfänger. Viele liessen sich durch anfängliche Probleme entmutigen. Heute ärgern sich alle über die Zeit, die sie durch falsche Scheu verloren haben.*

# Kapitel 3: Die Shell

## Übersicht

Du möchtest nun sicher einmal etwas am Computer selber tun. In diesem Kapitel lernst du das wichtigste über deinen direkten ”Ansprechpartner”. Dies ist nämlich nicht UNIX, sondern ein Programm, welches man Shell nennt.

### Lernziele

In diesem Kapitel lernst du:

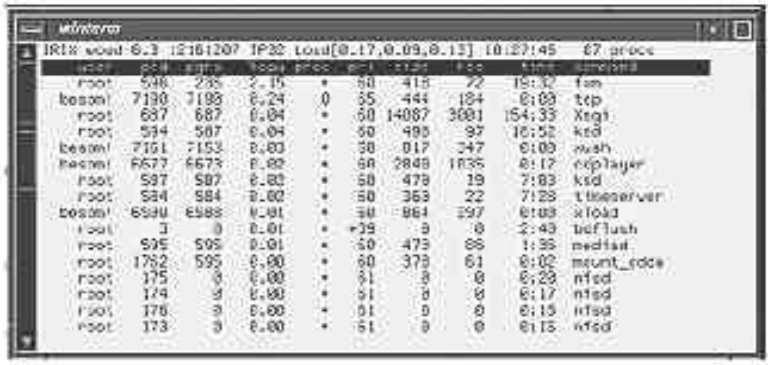
- wie UNIX-Kommandos aufgebaut sind
- die ersten Befehle kennen
- wie man sich in UNIX selber weiterhelfen kann

## Theorie

Im folgenden Abschnitt wirst du einiges über die Geheimnisse deines Dialogpartners erfahren. Die Interaktion geschieht nämlich nicht direkt mit dem Betriebssystem selber, sondern mit einer Schale, welche den Kern des Betriebssystems umfasst. Von da kommt auch der Name. Im Englischen bedeutet Shell Muschel oder Schale. Im folgenden wollen wir einige Begriffe als Synonym betrachten. Für die ersten Arbeiten ist diese Betrachtungsweise völlig legitim:

**Shell @Interaktionspartner @Betriebssystem UNIX @Fenster auf dem Bildschirm @ Kommando- oder Eingabefenster @DOS-ähnliches, kommandolinienbasiertes Eingabesystem.**


**Aufbau der Shell**



**Shellfenster in einem GUI**

UNIX-Benutzer nutzen heute die Vorteile einer graphischen Oberfläche genauso, wie sie nach wie vor Kommandos in die Shell eingeben. Der persönliche Arbeitsstil entwickelt sich meist zu einer Kombination aus den beiden Möglichkeiten.

# Arbeit für die Shell, die ersten Befehle

Um mit den UNIX Systembefehlen arbeiten zu können, musst du zuerst eine Shell starten. Am Anfang der Zeile steht oft eine Kombination aus Benutzer- und Rechnername, evtl. durch einen Punkt getrennt. Dann folgt meistens ein Doppelpunkt. Nach diesem wird angezeigt wo man sich im Filesystem befindet. Vor dem blinkenden Cursor findet sich dann noch ein Sonderzeichen. Je nach Umgebung kann dies eines der folgenden Zeichen sein: ‘#’, ‘>’, ‘%’. Die ganze Zeile wird Eingabeprompt oder kurz Prompt genannt.

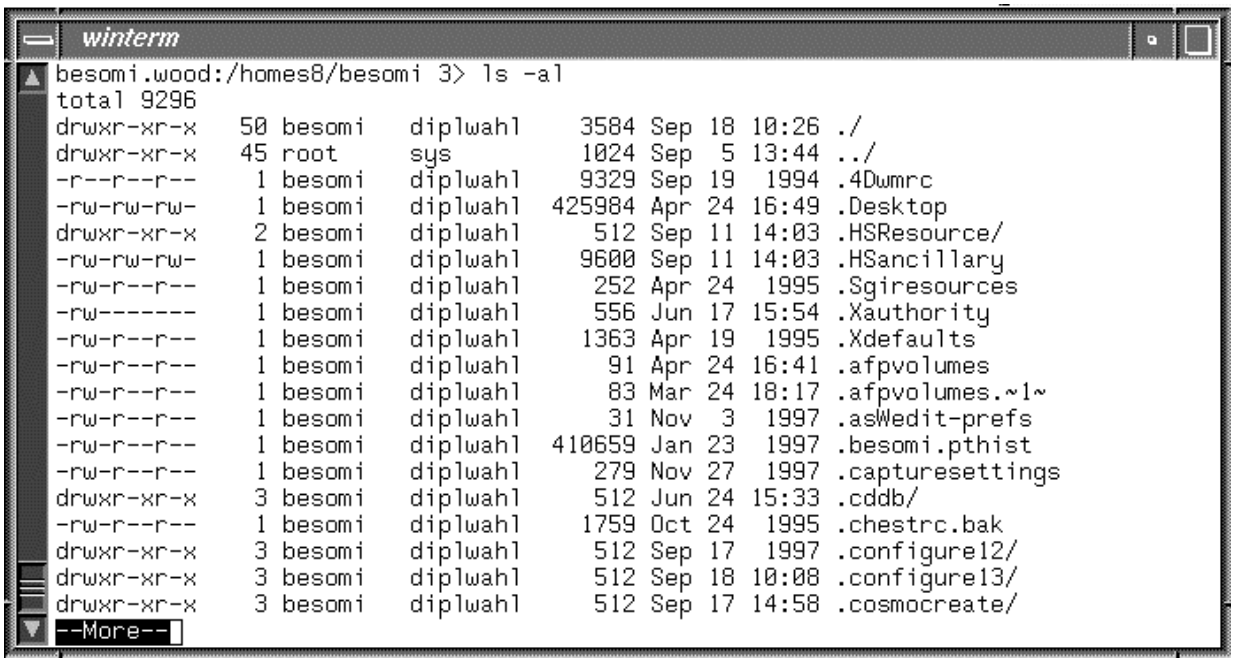
Schon bald wirst du Cristina über die Schultern schauen. Sie wird mit Häfi, dem virtuellen Männlein in ihrer Maschine, kommunizieren. Er verkörpert die Shell in diesem Leitprogramm. Der Prompt ist dabei wie folgt aufgebaut:
(Rechnername).(Benutzername):/(aktuelles Verzeichnis) % (blinkender Cursor).

```
compi.Cristina:/home%  |
```

In der Folge werden wir uns hauptsächlich damit befassen, mit welchen nützlichen Kommandos man die Shell füttern kann und was dabei herauskommt. Auf die Funktionalität, welche deine graphische Oberfläche noch bietet, werden wir nicht eingehen. Einen neuen Befehl wird fortan zuerst in einem Kasten dargestellt. Darin sollte ersichtlich sein, nach welchen Regeln das betreffende Kommando aufgerufen werden soll. Man nennt dies auch die Syntax eines Befehls.

```
Kommando [-Option] [Parameter]
```

Ein Beispiel aus der Praxis: Das Kommando, welches eingeben wurde, heisst **ls** (wir werden es später noch kennenlernen). Als Option wurde **-al** mitgegeben. Eine Eingabe wird jeweils mit Return bestätigt.



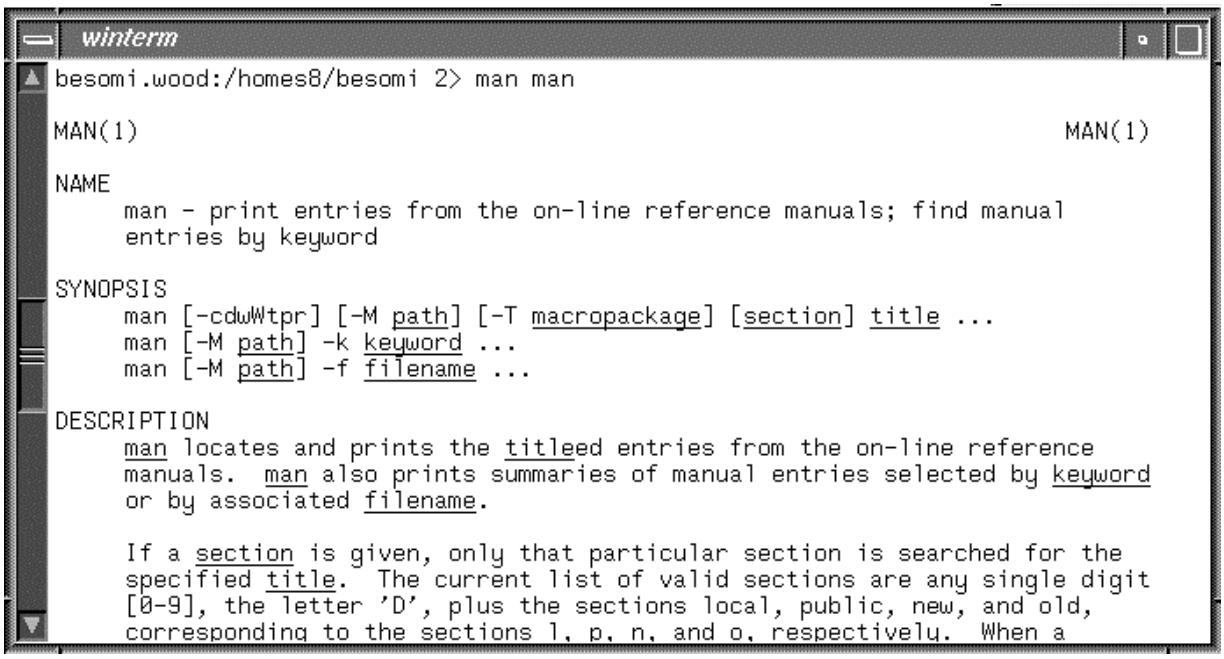

**Es gilt folgendes zu beachten:**

- UNIX unterscheidet Gross- und Kleinbuchstaben!
- Die Shell nimmt das erste Wort der Zeile als Kommando und sucht das zugehörige Programm. Der Rest der Zeile wird dem Programm als Anweisung mitgegeben. Zwischen einzelnen Anweisungsteilen (z.B. Kommando und Option) muss mindestens ein Leerzeichen stehen.
- Optionen haben immer ein vorangestelltes ”-” Zeichen.
- Eckige Klammern [ ] um einen Zusatz bedeuten, dass dieser optional ist. In der obigen allgemeinen Syntax siehst du, dass es Befehle gibt, die weder eine Option noch ein Parameter brauchen.

Mit der Zeit wirst du eine ganze Handvoll Befehle kennenlernen. Es würde zu weit führen, hier zu einem Kommando jeweils alle möglichen Optionen und Parameter zu erklären. Dies ist aber auch gar nicht nötig! UNIX stellt uns nämlich eine passende Hilfe zur Verfügung. Dies geschieht über ein Manual, welches es zu jedem existierenden Kommando gibt. Aufgerufen wird es wiederum mit einem eigenen Befehl:

| **man** ***Kommando*** |
|---|

Das Kommando **man** steht für manual. Will man zum Beispiel zusätzliche Informationen zum Kommando **ls** erhalten, so tippt man einfach **man ls** ein. Probier es doch aus! Wie du nachstehend siehst, kann man so auch über den Befehl **man** selber Informationen bekommen.



# Aufgaben

**Aufgabe 1**
Suche zuerst das Fenster mit deiner Shell. Es empfiehlt sich das erhaltene Passwort zu ändern. Dafür gibt es natürlich einen Befehl:

| **passwd** |
|---|

Obiges Kästchen sagt dir, dass das Kommando zum ändern des Passwortes **passwd** ist. Optionen wie allfällige Paramter scheint es nicht zu geben. Nach der Eigabe des Kommandos **passwd** muss man einfach den Instruktionen der Shell folgen.

**Aufgabe 2**
Du möchtest von der Shell das aktuelle Datum samt Uhrzeit erfahren. Hier der zugehörige Befehl:

| **date [Option] [Formatangaben]** |
|---|

Mit dem **date** -Kommando kann man bereits erstaunlich viel tun! Versuche deine Eingaben nun so zu machen, dass du als Ausgabe folgendes erhälst:
a) *Heute haben wir den 25.8.98 und es ist genau 17:35:11 Uhr*
b) *Den Wochentag von Weihnachten in diesem Jahr*
Tip: Mehr Informationen erhälst du mittels: **man date**

hier soll das aktuelle Datum und die aktuelle Uhrzeit stehen.

# Lösungen

1. passwd ....
2. date ”+Heute haben wir den %d.%m.%y und es ist genau %H:%M:%S Uhr”
3. date -d ‘dec 25 1998’ + ”%A”

# Kapitel 4: Das Filesystem

## Übersicht

**Die Organisation der Daten: bist du ein ordentlicher oder unordentlicher Mensch?**
Dieses Kapitel behandelt die Organisation deiner Daten, die sich im Computer befinden. Der Stoff ist so ausgelegt, dass du ihn direkt am Computer nachvollziehen kannst. Es ist wie bei dir zu Hause: du hast Schränke und Ordner, in welche du deine Dokumente ablegst. Du wirst sehen wie man solche Schränke erstellt und wie du deine Dokumente darin ablegen kannst. Schlussendlich wirst du dich so wohlfühlen wie bei dir zu Hause.

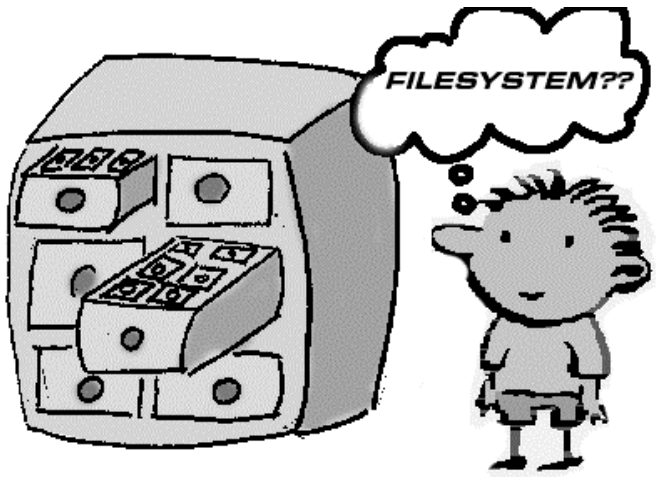
**Was tust du?**
Du befindest dich in der Welt von Cristina und Häfi, dem kleinen Menschen, der in ihrem Computer wohnt. Cristina versucht mit seiner Hilfe aufzuräumen, da bei ihr ein richtiges Chaos herrscht. Sie erklärt ihm, wie er ihre Daten organisieren soll. Du schaust den beiden über die Schulter. An deinem Computer versuchst du die Handlungen von Cristina nachzuvollziehen. Weiter schaust du, ob du die selben Antworten zurückbekommst wie sie Häfi an Cristina gibt. So wirst du hoffentlich von beiden eine Menge lernen.

## Lernziele

- Du kannst deine persönlichen Daten (z.B. die Ergebnisse deiner Arbeit) so behandeln, dass diese sicher sind vor unerwünschten Veränderungen oder einem versehentlichen Löschen, sowie, dass sie wiederauffindbar bleiben.
- Du kennst den Unterschied zwischen einem Directory (Verzeichnis), einem Homedirectory und einem File.
- Du kennst die wichtigsten Befehle, um dich durch den Datendschungel zu manövrieren.

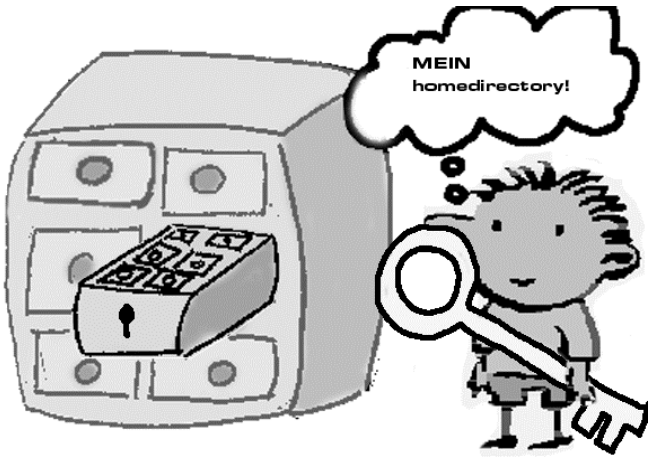
## Theorie: Datenverwaltung

Alle Daten, also Programme wie auch vom Benutzer erzeugte Daten, werden als Files in einem System gespeichert. Dieses System ist hierarchisch organisiert und heisst Filesystem.




Das Filesystem ist ein grosser Schrank. Dieser Schrank hat viele Schubladen. In diesen Schubladen kann man zusätzliche Fächer einrichten, die wiederum unterteilt werden können. So ist es möglich, sinnvoll seine Daten zu organisieren.

Unter UNIX hat jedes File einen Besitzer. Das heisst, es gehört einem bestimmten Benutzer. Nur dieser Benutzer darf ohne Einschränkungen mit diesem File arbeiten und kann auch bestimmen, in welchem Masse andere Benutzer Zugriff auf das File erhalten.




Ein Directory ist mit einer Schublade vergleichbar. Im Schrank (Filesystem) können verschiedene Leute eine entsprechende Schublade "mieten", in welchen sie ihre Dokumente aufbewahren und verwalten können. Die gemietete Hauptschublade heisst **Homedirectory**. Der Besitzer hat einen eigenen Schlüssel, um die Schublade zu öffnen.

- **Es wird dringend empfohlen, die Daten möglichst übersichtlich in Verzeichnissen zu organisieren! Dies erleichtert die Verwaltung der Daten erheblich.**
- **Alle Operationen die in der Shell ausgeführt werden, beziehen sich immer auf den Inhalt dieses Verzeichnisses (Directory).**

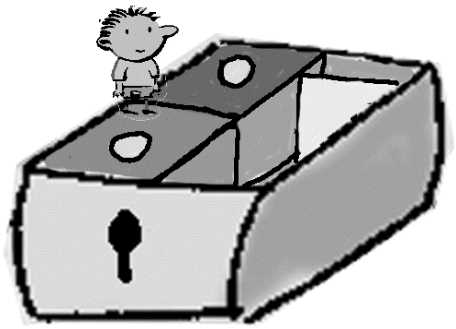
# Die Praxis

## Wo bin ich?

Hinter dem Prompt stehen fett die Befehle, welche Cristina in ihrer Shell eingibt. Darunter erscheinen jeweils die Ausgaben, die ihr Häfi zurücksendet. Nach dem Doppelpunkt im Eingabeprompt, beginnt die Beschreibung der aktuellen Position des Filesystems.

```
compi.Cristina:/home% pwd
/home
compi.Cristina:/home%  ls
hobby/          Bilder/          aufsatz.txt
compi.Cristina:/home%  |
```

**pwd** steht für **p**rint **w**orking **d**irectory
Damit hat Cristina gesehen, wo sie sich im Verzeichnisbaum befindet. Nach dem Aufstarten ist dies immer das eigene Homedirectory. Bei ihr heisst es */home* .
Mit **ls** (**l**i**s**t) hat sie dann Häfi gefragt, was sich in der aktuellen Schublade befindet. Häfi hat ihr darauf eine Liste zusammengestellt mit den Namen der Schubladen (directories) und der Dokumente (files). Wenn es sich um ein Verzeichnis (directory) handelt, schreibt Häfi hinter dem Namen ein "/". Handelt es sich um ein Dokument, schreibt Häfi einfach den entsprechenden Namen hin. Wie du siehst, befanden sich in ihrem Homedirectory die beiden Directories *hobby/* und *Bilder/*. Zusätzlich fand Häfi noch ein Datei mit dem Namen *aufsatz.txt*.



Du hast also zwei neue Befehle kennengelernt:
Probier sie in deiner Shell aus!

```
pwd
```

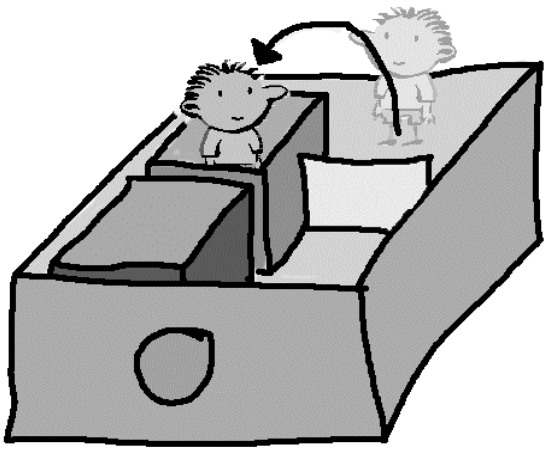
```
ls
```

### Wie öffne ich eine andere Schublade?

Bei Cristina ging es folgendermassen weiter:

```
compi.Cristina:/home% cd hobby
compi.Cristina:/home/hobby% ls
meinvelo.ps              veltour98.txt
compi.Cristina:/home/hobby% cd ..
compi.Cristina:/home% pwd
compi.Cristina:/home%  |
```

**cd** steht für **c**hange **d**irectory.
Mit **cd hobby** wollte Cristina in die Unterschublade *hobby/* wechseln, was Häfi auch sofort für sie erledigt hat. Im neuen Eingabepromt bestätigt dies Häfi mit der Angabe */home/hobby*, als aktuelles Directory. Das Directory *hobby/* ist in der Hierarchie eine Stufe unter dem Homedirectory */home/*. Man nennt es deshalb auch ein Subdirectory oder auf Deutsch ein Unterverzeichnis. Was **ls** tut, weißt du ja bereits. Mit **cd ..** wechselte Cristina dann wieder eine Hierarchie-Stufe hinauf. Zur Kontrolle hat sie mit **pwd** nochmals nachgefragt, ob sie sich jetzt wieder im Homedirectory befindet. Häfi hat seine Arbeit richtig gemacht und ihr die erwartete Bestätigung ausgegeben.



Was hast du hier gelernt:

**cd** *Directoryname*

Anstelle des Directorynamens kann man auch zwei Punkte angeben. So wechselt man hierarchisch gesehen ins nächsthöhere Verzeichnis.

### Wie erstelle und entferne ich Schubladen?

Zwei weitere wichtige Kommandos helfen bei dieser Frage weiter:

| | |
|---|---|
| **mkdir** *Directoryname* | Mit **mkdir** wird ein neues Directory erzeugt (**m**a**k**e **dir**ectory). |
| **rm** *Name* | **rm** kommt von **r**e**m**ove. Man kann damit Dateien und Directories löschen. |

### Aufgabe zur Lernkontrolle

Cristina hat Häfi bereits weiterbeschäftigt. Versuche zu verstehen was sie getan hat. Beschreibe in eigenen Worten Cristinas Vorgehen.

```
compi.Cristina:/home%  cd hobby
compi.Cristina:/home/hobby% mkdir tonnis
compi.Cristina:/home/hobby% mkdir velo
compi.Cristina:/home/hobby% ls
meinvelo.ps   tonnis/   veltour98.tx   velo/
compi.Cristina:/home/hobby% rm tonnis
compi.Cristina:/home/hobby% mkdir tennis
compi.Cristina:/home/hobby% cd ..
compi.Cristina:/home%
```

### Wie verschiebe oder kopiere ich Dokumente?

Im Englischen heisst bewegen move. Daher der Befehl **mv,** für das Verschieben von Dateien

**mv** *Name Ziel*

Der Befehl **cp** kommt von **c**o**p**y.

**cp** *Name Ziel*

*Name* ist ein File- oder Directory-Name aus dem aktuellen Directory. *Ziel* kann ein beliebiges Directory sein. Wenn Daten nicht verschoben, sondern kopiert werden sollen, verwendet man **cp**. Beachte, dass zum Kopieren von Ordnern die Option **-r** angegeben werden muss, also:

**cp -r** *Name Ziel*

Weiter sei darauf hingewiesen, dass man mit **mv** auch Datei- oder Directory-Namen umbenennen kann. Die Syntax lautet dann folgendermassen.

**mv** *alterName neuerName*

Cristina hat folgendes getan:

```
compi.Cristina:/home% cd hobby
compi.Cristina:/home/hobby% ls
meinvelo.ps      tennis/    velotour98.txt velo/
compi.Cristina:/home/hobby% mv meinvelo.ps /home/hobby/velo/
compi.Cristina:/home/hobby% mv velotour98.txt  /home/hobby/velo/
compi.Cristina:/home/hobby/velo% cd velo
compi.Cristina:/home/hobby/velo% mv velotour98.txt tour98.txt
compi.Cristina:/home/hobby/velo% ls
```

Zuerst wechselt Cristina ins Verzeichnis *hobby* und lässt sie den Inhalt anzeigen. Danach hat sie Häfi zwei Verschiebungsaufträge gegeben. Er hat der Reihe nach die beiden Files *meinvelo.ps* und *velotour98.txt* ins Unterverzeichnis *velo* verschoben. Danach der Wechsel nach *velo* und die Umbenennung von *velotour98.txt*. Zur Kontrolle nochmals ein **ls**.

## Aufgabe

Cristina möchte nun endlich ihre Dokumente von der Schule sinnvoll ordnen. Folgende Dokumente liegen im Moment im Home-Verzeichnis:

| **Filename** | **Beschreibung** |
|---|---|
| essay01.txt | Englisch-Aufsatz |
| essay02.txt | Englisch-Aufsatz |
| biovortrag.doc | Biologievortrag |
| myhp.html | eigene Homepage |
| mathe1zus.doc | Zusammenfassung des ersten Semesters in Mathematik |
| 98ueb1.ms | Übung 1 in Mathematik 1998 |
| 98ueb2.doc | Übung 2 in Mathematik 1998 |
| 98ueb3.doc | Übung 3 in Mathematik 1998 |
| SREngadin.tif | Klassenbild der Schulreise |
| PlanSS98.doc | Stundenplan des Sommersemester 1998 |
| matheprüf.doc | Alte Mathematikprüfrungen |

a) Wie würdest du die Directories anordnen? Zeichne von Hand eine Verzeichnisstruktur. Du kannst dies anhand eines Baumes darstellen oder einer anderen geeigneten Form.

b) Du hast dich in Teilaufgabe a) für eine Hierarchie entschieden. Setzt nun deinen Vorschlag am Computer um.
Alternative: Wenn du bereits eine klare Vorstellung davon hast, welche Ordner du für deine Arbeit brauchen wirst, so kannst du auch dazu die entsprechenden Verzeichnisse erstellen.

## Lösung

a) Ein Vorschlag wäre der folgende. Natürlich gibt es auch andere Möglichkeiten. Die weissen Rechtecke entsprechen Directories, die Grauen den, in der Aufgabe erwähnten Files.

b) + c) Diese Directory-Struktur erstellt man durch wiederholtes Anwenden der Kommandos **mkdir** und **cd**. Für den Pfad Mathe/ ist das Vorgehen hier nochmals aufgezeigt.

```
compi.Cristina:/home% mkdir Schule/
compi.Cristina:/home% cd Schule
compi.Cristina:/home/Schule% mkdir Mathe/
compi.Cristina:/home/Schule/Mathe% cd Mathe
compi.Cristina:/home/Schule/Mathe% mkdir Vorlesung
compi.Cristina:/home/Schule/Mathe% mkdir Uebungen
compi.Cristina:/home/Schule/Mathe% mkdir Prüfungen
compi.Cristina:/home/Schule/Mathe% cd ..
compi.Cristina:/home/Schule% cd ..
compi.Cristina:/home% mkdir English
```

# Kapitel 5: Wichtige Kleinigkeiten

## Übersicht

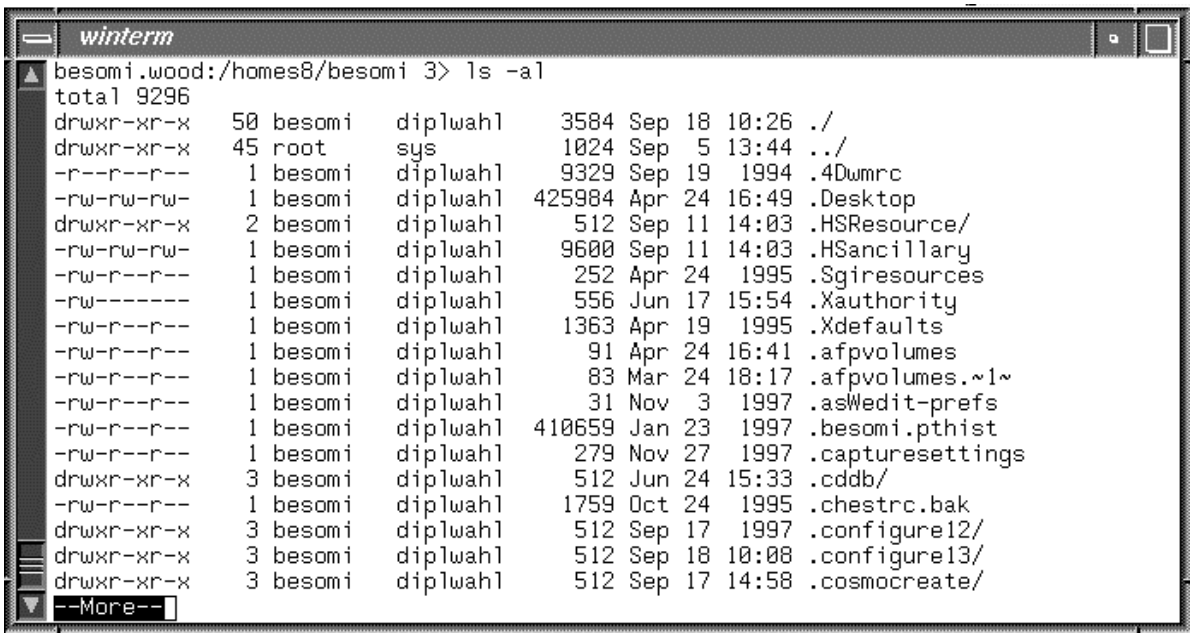
Dieses Kapitel ergänzt dein Wissen über die Handhabung deines Filesystems. In zwei kurzen Abschnitten wirst du hilfreiche Details kennenlernen, die dir die tägliche Arbeit erleichtern werden.

### Lernziele

- Du bist in der Lage, deine Files oder Directories vor fremdem Zugriff zu schützen.
- Du bist vertraut mit zwei einfachen Wildcards. Die werden in Zukunft u.a. beim Wiederauffinden von Dokumenten behilflich sein.

### Theorie: Fileattribute

Jede Datei und jedes Directory besitzt unter UNIX nicht nur einen Namen sondern noch eine ganze Reihe weiterer Informationen. Diese spezifischen Merkmale (Attribute) sind im Datei-kopf gespeichtert. Gibt man beim **ls**-Kommando die Option **-l** an, so erhält man viele dieser Attribute angezeigt. Falls das aktuelle Directory, in dem man sich befindet, nicht leer ist, kann es dann in der Shell etwa so aussehen:



Wir wollen uns nur mit dem ersten Block befassen, der aus 10 Zeichen besteht. Im obigen Bild heisst dieser Block der ersten Ausgabezeile: ***drwxr-xr-x***
Der erste Buchstaben gibt an, um welches Objekt es sich handelt. Steht ein ***'d'***, so handelt es sich um ein Directory. Steht ein '- ' so ist es eine normale Datei. Alle anderen Buchstaben die vorkommen können (c,b,p) kennzeichnen spezielle Dateitypen. Allgemein gesagt beschreibt diese Position den Dateityp (ein Directory wird dabei auch als eine Art Datei angeschaut).

Die Zeichen auf den Positionen 1-9 regeln die Zugriffsrechte auf das File bzw. das Directory und haben folgende Bedeutung:

- **r** steht für readable (lesbar)
- **w** steht für writeable (schreibbar und löschbar)
- **x** steht für executable (ausführbar). Dies ist dann wichtig, wenn es sich bei der Datei um ein Programm handelt. Bei einem Directory bedeutet das **x**, dass man mit cd in das Verzeichnis hinein wechseln kann.
- \- kein Recht auf dieser Position

Die 9 Zeichen sind in drei Dreiergruppen unterteilt. Die ersten drei Positionen beschreiben die Rechte des Besitzers. Positionen 5-7 regeln analog die Zugriffsrechte einer Gruppe. So z.B. die Gruppe aller Studenten an einer Abteilung. Zum Schluss folgt dasselbe für andere aussenstehenden Personen, die ebenfalls Zugriff auf das Netzwerk haben. Man erhält folgenden Übersicht:

| **Pos.** | **... gibt Antwort auf die Frage:** | **Bsp.** |
|---|---|---|
| 1 | um welchen Dateiityp handelt es sich? | d |
| 2 | Hat der Besitzerr das Leserecht? | r |
| 3 | Hat der Besitzer das Schreibrecht? | w |
| 4 | Hat der Besitzer die Ausführerlaubnis? | x |
| 5 | Haben Leute der gleichen Benutzergruppe das Leserecht? | r |
| 6 | Haben Leute der gleichen Benutzergruppe das Schreibrecht? | - |
| 7 | Haben Leute der gleichen Benutzergruppe die Ausführerlaubnis? | x |
| 8 | Haben andere aussenstehende Personen das Leserecht? | r |
| 9 | Haben andere aussenstehende Personen das Schreibrecht? | - |
| 10 | Haben andere aussenstehende Personen die Ausführerlaubnis? | x |

Es versteht sich, dass nur der Eigentümer einer Datei auch deren Zugriffsrechte ändern kann. Wenn immer du in deinem Account ein File oder ein Directory erzeugst, werden die Attribute automatisch wie folgt gesetzt:

Directory: drwxr-xr-x
File: -rw-r--r—

Das bedeutet, dass jedermann deine Directories einsehen, die Files lesen und die Programme ausführen kann. Auch neu erzeugte Files sind der Öffentlichkeit zur Einsicht freigegeben. Dies entspricht der üblichen offenen Philosophie von UNIX.

# Theorie: Wildcards

Es gibt in UNIX verschiedene Zeichen die man als Platzhalter benutzen kann. Man nennt diese Zeichen auch Wildcards oder Metazeichen. Diese Metazeichen haben alle eine spezielle Bedeutung. Die wichtigsten Zeichen sind dabei:

| **Zeichen** | **Bedeutung** |
|---|---|
| * | Steht für beliebig viele (auch Null) Zeichen. |
| ? | Ist Platzhalter für genau ein einzelnes beliebiges Zeichen (aber nicht für ein Leerzeichen) |

Hier einige Beispiele wie man solche Wildcards gebrauchen kann:

| | |
|---|---|
| ls .* | Listet alle Directoryeinträge auf, die mit einem Punkt beginnen. |
| ls *.txt | Listet alle Textfiles auf (wegen der Endung txt). |
| ls lösung??.doc | Alle doc-Files, deren Name mit lösung beginnt und danach noch zwei Zeichen folgen, werden aufgelistet. |

# Aufgaben

**Aufgabe 1**

a) Erstelle ein neues Directory mit dem Namen ”Privat”. Wie sehen nun die Zugriffsrechte für die verschiedenen Benutzergruppen aus?
b) Ändere nun die Zugriffsrechte so, dass nur du in dieses Directory einsehen kannst. Das Kommando mit dessen Hilfe du dies bewerkstelligen kannst, heisst **chmod**. Zur Lösung dieser Aufgabe musst du dir das Manual zu diesem Kommando anschauen.


**Aufgabe 2**

Cristina verkauft nebenbei selber gemachte Wandteppiche in der ganzen Welt. Jeden Monat hat sie mehr begeisterte Kunden gewonnen. Alle Dokumente die sie bis jetzt hat, befinden sich in ihrem Directory Kunden. Jedes dieser Files beginnt mit dem Kundennamen. Da sich nun bereits über 100 Dateien in diesem Ordner befinden, möchte sie Ordnung schaffen.
Schlage ihr eine einfache Lösung vor, wie sie alle ihre Dokumente der Kunden, die mit ‘a’ beginnen in einen Ordner ‘A’ schieben kann.

a) Erstelle erst einen Subdirectory ‘A’ im Ordner Kunden.
b) Verschiebe alle Files von Kunden die mit ‘a’ beginnen nach ‘A’.

# Lösungen

**Aufgabe 1**

a+b) Die Zugriffsrechte sehen folgendermassen aus: drwxr-xr-x
Das heisst, dass jedermann die Directories einsehen, die Files lesen und die Programme ausführen kann. Für das Ändern der Zugriffsrechte gibt es zwei Möglichkeiten:

```
compi.Cristina:/home% mkdir Privat
compi.Cristina:/home% chmod go-x Privat/
compi.Cristina:/home%>  |
```

```
compi.Cristina:/home% mkdir Privat
compi.Cristina:/home% chmod 700 Privat/
compi.Cristina:/home%>  |
```

**Aufgabe 2**

Die Befehle würden z.B. so aussehen:

```
compi.Cristina:/home%> mkdir Kunden
compi.Cristina:/home%> cd Kunden
compi.Cristina:/home/Kunden%> mkdir A
compi.Cristina:/home/Kunden%> cd Kunden
compi.Cristina:/home%> mv ../a* .
compi.Cristina:/home%>  |
```

# Additum: Aufstarten von Programmen

## Übersicht

Auf einem Datenträger sind verschiedene Arten von Files abgespeichert. Man unterscheidet in erster Linie zwischen Daten-Files (Texte, Bilder, Tabellen etc.) und sogenannten executable Files (Programme, auch Applikationen genannt). Wir wollen sehen, wie man solche Programme startet.

### Lernziele

- Du lernst, wie man Programme startet und wie man sie wieder beenden kann.
- Du weißt, wie man Programme im Hintergrund laufen lässt.

## Wie starte ich ein Programm?

Um ein Programm aufzustarten, tippt man in der Shell einfach den Namen des Programms ein. Das Programm muss entweder im aktuellen oder in einem vom Systemadministrator dafür vorgesehenes Directory enthalten sein. Häfi geht dabei also folgendermassen vor:
Zuerst schaut er, ob der Befehl an ihn direkt gerichtet ist. Wenn er nichts findet, dann schaut er, ob es so ein File im aktuellen Directory gibt. Findet er da auch nichts, so schaut er im Filesystem weiter, ob der Systemadministrator ein solches Programm installiert hat.

### Lernaufgabe

Erkundige dich was für Programme dir zur Verfügung stehen.

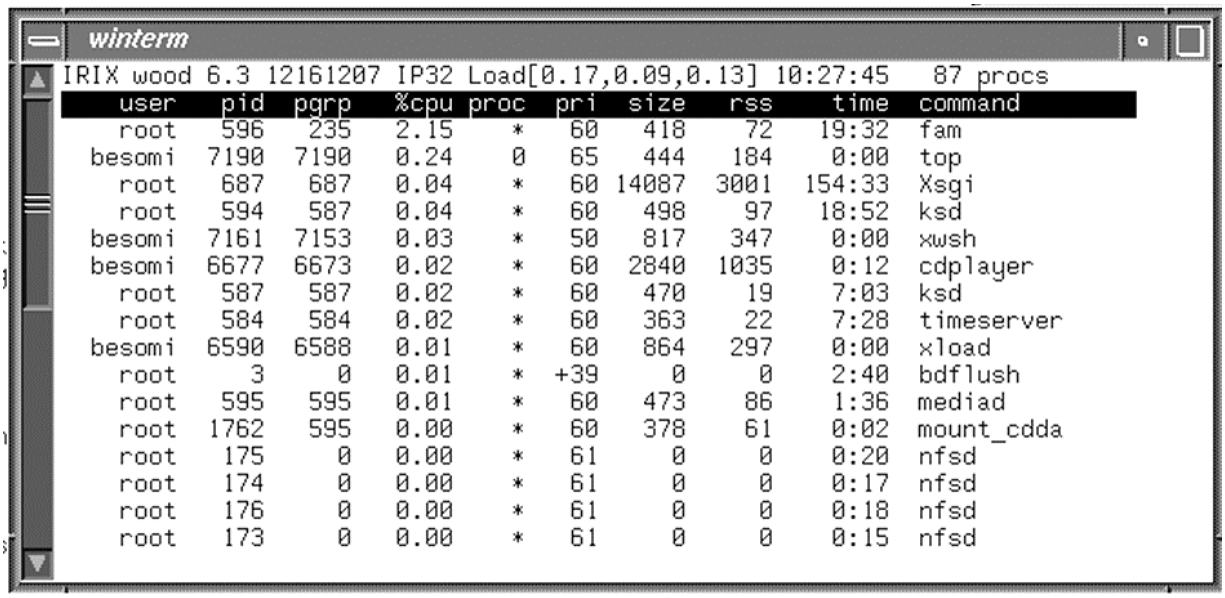
a) Es gibt zahlreiche Texteditoren für UNIX. Es würde zu weit führen hier näher auf einen einzelnen einzugehen. Erkundige dich bei einem anderen Benutzer oder bei deinem Lehrer, ob er einen Editor empfehlen kann. Meistens wird heute ein Editor verwendet, der in der graphischen Oberfläche integriert ist.
b) Erkundige dich, ob du die Möglichkeit hast, das Internet zu nutzen. Vielerorts ist heute der Browser von Netscape installiert. Probier aus, ob du ihn starten kannst, indem du in der Shell **netscape** eintippst. Ansonsten musst du auch da nachfragen.

## Programme im Hintergrund laufen lassen.

Da die Shell selber auch nur ein Programm ist, kann man in verschiedenen Fenstern gleichzeitig Befehle an das System senden. Man kann auch mehrere Shells gleichzeitig geöffnet haben (mehrere Fenster; nur in einem GUI möglich). Diese Möglichkeit mag am Anfang etwas verwirrend sein, gehört aber zum Alltag eines jeden UNIX Benutzers. Hat man aber in einer Shell ein Programm aufgerufen, so bleibt die Shell "hängen", bis dieses beendet wird. Hängen bleiben bedeut, dass du keine neuen Befehle eingeben kannst, da kein neuer Eingabeprompt erscheint. Dies kann verhindert werden, indem man das Programm im Hintergrund laufen lässt. Dies wird erreicht, indem man dem Kommando ein "&" anhängt.

## Welche Programme laufen gerade? Wie beende ich ein Programm?

Unter Unix laufen immer eine ganze Menge von Programmen (Prozessen) gleichzeitig. Mit den Befehl **top** kann man sich die Sache ansehen. Der Befehl top sortiert die Prozesse nach dem Grad der Belastung des Computers.




Es kann nun vorkommen, dass ein Programm abgestürzt ist. Es reagiert nicht mehr, oder man weiss nicht, ob sich noch irgend etwas tut. In solchen Fällen kann man das Programm mit mehr oder weniger Gewalt abbrechen. Vielfach kommt man wieder heraus, indem man die Kontroll-Taste gemeinsam mit dem Buchstaben c oder z drückt (<CTRL+c> oder <CTRL+z>). Hilft dies nicht weiter, so lässt man sich mit **top** die Liste der aktuellen Prozesse angeben. Darin sucht man den Prozess, den man gerne abbrechen würde und merkt sich in der 2. Spalte die Prozessnummer (steht unter pid, was so viel heisst wie process-id). Danach kann man mit dem Befehl **kill** den Prozess beenden.

**kill [-9]** ***Prozessnummer(n)***

Verwendet man zusätzlich die Option **-9,** dann wird der Prozess 100% abgeschossen! Meistens geht’s aber auch ohne.

# Anhang: wichtige UNIX- Befehle

| | |
|---|---|
| man command | Beschreibung des Kommandos |
| ls [-l] [ file] | Liste der Directory-Einträge |
| cp [-i] fil1 fil2 | Kopiere fil1 nach fil2 |
| mv quelle ziel | Datei umbenennen  bzw. verschieben |
| rm [-i] file | Entferne Datei file |
| pwd | Aktuelles Verzeichnis anzeigen |
| cd dir | Directory wechseln (DOS cd, VMS set def) |
| mkdir dir | Directory anlegen |
| rmdir dir | Directory entfernen |
| cat file | Textdatei anzeigen |
| more file | Textdatei seitenweise anzeigen |
| find file -print | Finde eine Datei |
| diff fil1 fil2 | Unterschiede zwischen fil1, fil2 |
| grep string file | Suche in file nach string |
| vi file | Editiere file (vorher vi-Kommandos lernen!) |
| ln [-s] fil1 fil2 | [symbolischer] Link fil1 mit fil2 |
| chmod nnn file | Zugriffsrechte ändern |
| umask nnn | (csh) Default-Rechte setzen |
| cc [...] file.c | C-Compiler aufrufen |
| file | ausführbares Programm starten |
| file & | Programm als Hintergrundprozeß starten |
| time file | Programm starten, CPU-Zeit stoppen |
| ^C | Programm beenden |
| ^Z | Programm vorübergehend stoppen |
| fg | Programm in den Vordergrund holen |
| bg | Programm im Hintergrund laufen lassen |
| jobs | Information über Hintergrund-Prozesse |
| kill [-...] pid | Prozeß beenden |
| exit | Shell oder Skript beenden |
| alias nam command | "Alias" definieren |
| env | Umgebungsvariablen anzeigen |
| setenv ENVPAR value | Umgebungsvariable setzen |
| set param=value | Shell-Variable setzen |
| echo $param | Variable anzeigen |
| echo string | Gib string aus |
| passwd | Ändere das Paßwort |
| telnet host | Interaktives Login auf host |
| ftp host | Dateitransfer |
| netstat | Netzwerk-Nutzung |
| stty | Terminaleinstellungen |
| date | Datum und Zeit anzeigen |
| who | Benutzer dieses Systems |
| rwho | Benutzer entfernter Systeme |
| ps | Prozeß-Informationen (VMS show system) |
| df | Information über Dateisysteme |
| lpr -Pque file | Drucke file auf Queue que |
| lpq | Drucke Warteschlangen-Status |
| lp -dDest file | Drucke file auf Queue Dest |

# Literatur

Unix (ETH-Z, Architecture & CAAD)
http://alterego.arch.ethz.ch/infolan/info/tutorials/Unix/

Wichtige UNIX Commands (ETH-Z, Department of Electrical Engineering)
http://www.ee.ethz.ch/isg/tardis/commands/

Kurzeinführung in UNIX (FU Berlin, Fachbereich Chemie)
http://www.chemie.fu-berlin.de/chemnet/general/unix-inf.html

Grundsätzliches über UNIX (Ulm)
http://donau.ul.bawue.de/IN/help/unix/anl_2.html

UNIXhelp for Users (University of Edinburgh)
http://unixhelp.ed.ac.uk/index.html

UNIX commands >> Suche mit +unix +commands in AltaVista
http://altavista.digital.com/