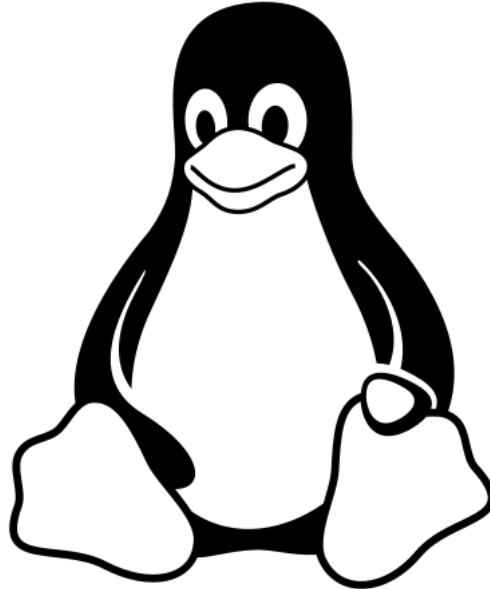


SKRIPT ZUM UNIX/LINUX-CRASHKURS

UNIVERSITÄT TRIER FB IV FACHSCHAFT INFORMATIK



Liebe Erstsemester,
willkommen an der Uni Trier! Wir, der Fachschaftsrat Informatik, helfen euch gerne dabei richtig ins Studium zu starten. Bevor wir anfangen, wollen wir kurz den Begriff Fachschaftsrat erklären, damit keine Missverständnisse aufkommen. Ein Fachschaftsrat (FSR) besteht aus ~ 10 Mitgliedern und ist die gewählte Studentenvertretung einer Fachschaft. Eine Fachschaft ist nichts anderes als alle Studierende eines Faches. Zum Beispiel bilden alle Informatikstudenten eine Fachschaft. Die Mitglieder eines FSRs setzen sich ehrenamtlich für die Interessen der jeweiligen Fachschaft ein. Konkret bedeutet dies, dass wir z. B. Skripte und **Klausurprotokolle** an euch weiterleiten, eure Druckerkonten aufladen oder auch Lanpartys organisieren. Sollten Probleme mit den Dozenten oder Professoren auftauchen, stehen wir euch zur Seite.

Dieses Skript versucht euch an das Betriebssystem Linux heranzuführen. Um dies zu unterstützen halten wir zu Beginn jedes Semesters einen Linux Crashkurs. Wir hoffen, dass dieser euch weiterhilft und sind auf eure Anmerkungen und Kritik gespannt.

Viel Spaß an der Uni! :)

wünscht euch euer Fachschaftsrat Informatik

Campus 2, Raum H 508 <http://fsrinfo.uni-trier.de> fsrinfo@uni-trier.de

1	Allgemeines	3
1.1	CIP-Pools	3
1.2	Drucksystem	3
1.3	Wichtige Punkte der Benutzerordnung	4
2	Einleitung	5
2.1	Woher stammt Linux?	5
2.2	Was ist Linux?	5
2.3	Warum Linux?	5
3	Einführung in Linux/Unix	7
3.1	Der Login	7
3.1.1	Passwort ändern	7
3.1.2	Passwörter sicher wählen	8
3.2	Bedienungsoberfläche	8
3.3	Die Shell (Kommandozeile, Terminal, Konsole)	8
3.3.1	Warum die Shell verwenden die graphische Oberfläche ist doch viel komfortabler?	9
3.3.2	Kommandos	9
3.4	Das Dateisystem	10
3.4.1	Navigieren im Verzeichnisbaum	10
3.5	Arbeiten mit Dateien und Verzeichnissen	11
3.5.1	Verzeichnisse anlegen und löschen	11
3.5.2	Dateien lesen, erstellen, kopieren und löschen	11
3.5.3	Verzeichnisse und Dateien verschieben	12
3.5.4	Verzeichnisse oder Dateien verknüpfen	12
3.5.5	Den belegten Speicherplatz ermitteln	12
3.5.6	Durchsuchen mit Grep	12
3.5.7	Finden mit Find	13
3.5.8	Komprimierte und archivierte Dateien	13
3.6	Das Leben mit dem Terminal	14
3.7	Datei- und Benutzerrechte	14
3.8	Hilfe	16
3.8.1	Manpages und die wichtigsten Befehle (man, whatis, apropos, -h, info)	16
3.8.2	Weitere Hilfe in der Dokumentation und im Internet	16
3.8.3	Admins fragen!	16
3.9	Prozesse (Prozess = Befehl, Programm)	17
3.9.1	Hintergrundprozesse	17
3.9.2	Signale	17
3.10	Ein- und Ausgabeumleitung	18
3.11	Drucken auf der Shell	18
3.12	Einfache reguläre Ausdrücke (Wildcards)	19
3.13	Netzwerk	19
3.13.1	Secure Shell	19
3.13.2	Secure Copy	20
3.13.3	Zugriff von Windows aus (PuTTY, FTP-Client, Cygwin)	20
3.14	Spielereien im CIP-Pool	21
4	Sonstiges...	22

1 Allgemeines

1.1 CIP-Pools

CIP-Pools sind Rechnerräume. Um die Rechner in den CIP-Pools der Informatik zu benutzen, benötigt man eine Zugangserlaubnis (Benutzeraccount), den man beim Administrator (Raum H 525) erhält. Dazu gehört eine E-Mail-Adresse der Form `name@stud.informatik.uni-trier.de` bzw. `name@castor.uni-trier.de`, an die auch wichtige Mitteilungen der Abteilung (Seminarankündigungen) oder des Fachschaftsrats geschickt werden. Jedes Semester muss man seinen Account im CIP-Pool H 523 rückmelden, sonst wird er deaktiviert.

Ein Forward kann mit dem Account-Tool (Menü → Einstellungen → CIP-Pool Useraccount) eingerichtet werden. Es ist auch möglich über die Konsole eine Weiterleitung einzurichten. Dazu öffnet man ein Konsolenfenster und wechselt mit dem Befehl `cd ~` ins erforderliche Verzeichnis. Dort erstellt man mit `gedit .forward` eine Textdatei und trägt in diese die E-Mail-Adresse ein, an die die Mails weitergeleitet werden sollen. Nach dem Abspeichern können Editor und Konsole geschlossen werden. Die Weiterleitung ist nun aktiv.

Den Studierenden stehen folgende CIP-Pool-Räume zur Verfügung:

- **Räume H 523 und H 524** CIP-Pools der Informatik
von den Admins der Informatik betrieben, Rechner mit Linux, siehe `cip.uni-trier.de`
- **Raum H 424** CIP-Pools der Wirtschaftsinformatik
vom ZIMK betrieben, mit Windows

Des weiteren betreibt das Rechenzentrum der Uni Trier (Zimk) in allen Gebäuden der Universität PC-Pools mit Windows z. B. auch in E 09 und F 58. Für diese Pools benötigt man einen Account vom Rechenzentrum. Damit verbunden ist eine weitere E-Mail-Adresse, an die wichtige und weniger wichtige Mitteilungen der Uni-Verwaltung, des AStA und zahlreicher studentischer Interessengruppen und Vereinigungen geschickt werden - hier empfiehlt es sich, einen guten Filter einzurichten. Weitere Informationen zum Zimk unter <http://zimk.uni-trier.de>. Auch das Fach Mathematik hat eigene Rechner, für die man in E 3 (E-Gebäude, Erdgeschoss) einen Zugang beantragen kann.

Sämtliche CIP-Pools an der Universität (auch die des ZIMK) sind verschlossen und müssen über die Keypads neben den Türen geöffnet werden. Dazu muss auf der Seite <https://zeus3.uni-trier.de> (bitte das „s“ von „https“ nicht vergessen!) eine PIN vergeben werden. Auf dieser Seite muss man sich mit seiner Universitätskennung (die man zum Login auf PORTA und StudIP nutzt) und dem zugehörigen Passwort einloggen und kann dann eine fünfstellige Zahl als PIN auswählen. Bitte beachten: Die Fehlererkennung ist nicht vollständig umgesetzt, sodass man (Stand November 2015) auch andere Symbole als Zahlen auswählen kann, wie beispielsweise „}“. Auf den numerischen Keypads lassen sich solche Symbole aber nicht eingeben, sodass man beachten sollte, dass die PIN auch ausschließlich Zahlen enthält. Die erstellte PIN gilt für alle PC-Pools der Universität.

1.2 Drucksystem

Jeder Benutzer hat ein eigenes Druckerkonto, über das gedruckte Seiten abgerechnet werden. Das Startguthaben beträgt 20 Seiten (s/w). Ihr könnt Euer Druckerkonto bei der Fachschaft Informatik (H 508) aufladen. Es ist meistens einer im Büro der Euch gerne weiterhilft. Euren Kontostand könnt Ihr unter <http://cip-print.stud.informatik.uni-trier.de> (nur uni-intern) einsehen. Dort könnt

Ihr auch die aktuellen Preise nachlesen. Es stehen ein S/W-Laserdrucker und zwei Farblaserdrucker (A3 und A4) in den CIP-Pools zur Verfügung. In H 524 stehen auch Tacker und Locher. (siehe ??)

1.3 Wichtige Punkte der Benutzerordnung

Jeder sollte vor dem ersten Nutzen der Rechenanlagen die Nutzerordnung genau durchlesen. Diese hängt in den Pools aus und ist auch im Systemadministratorenbüro H 525 einzusehen. Im Folgenden aber die wichtigsten Punkte:

- **Das Ausschalten der Geräte in den CIP-Pools ist nicht gestattet!**

- Die Geräte und Software leiden darunter.
- Rechner rebooten nicht. BIOS-Passwörter sind gesetzt, damit die Rechner nur von den Systemadministratoren neu gestartet werden können. Solange dieses Passwort nach dem Einschalten noch nicht eingegeben wurde, sind die Geräte nicht weiter verwendbar!
- Andere Benutzer werden gestört, unterbrochen oder verlieren Daten! Da an einem Rechner mehr als ein Benutzer arbeiten kann (über das Netz von einem anderen Rechner aus) wird dieser durch das ausschalten unterbrochen. Dabei kann er nicht gespeicherte Daten verlieren!

Wer Rechner ausschaltet oder neu startet, zieht nicht nur den Zorn der anderen Benutzer und der Systemadministratoren auf sich, er riskiert auch bei wiederholtem Verstoß gegen die Benutzerordnung seine Rechnererlaubnis!

Bildschirme dürfen abends, freitags vor dem Wochenende und in den Ferien ausgeschaltet werden. Dies spart nicht nur Strom, sondern verlängert auch deren Lebenszeit.

- **Falls Probleme mit einem Rechner auftreten: meldet diese den Systemadministratoren (H525)**

Es kommt immer wieder vor, dass ein und der selbe Rechner von irgendwem wegen irgendwelchen Problemen ausgeschaltet wird. Sysadmins können diese allerdings nicht beseitigen, wenn sie nie erfahren, was das für Probleme sind! (Wenn sie nicht da sind, eine E-Mail an root@stud.informatik.uni-trier.de schreiben oder eine Notiz unter die Tür schieben.)

- **Die Pools sind dem wissenschaftlichen Arbeiten vorbehalten!**

In den Hauptnutzungszeiten der Pools (ca. 10:00 - 16:00) ist das Spielen und das private Surfen im Internet untersagt! Zu jeder Zeit ist Nutzern, die die Rechner zum Arbeiten verwenden wollen, den Vorzug zu geben!

- **Herunterladen von Daten**

Das Herunterladen von größeren Datenmengen ist nur zu wissenschaftlichen Zwecken erlaubt. Private Daten (wie Videos, Musik, Programme, usw) herunterzuladen, ist strengstens verboten. Der Missbrauch der Universitätsleitungen wird in der Regel mit Entzug der Rechnerlaubnis bestraft. Richtwert: max. 2 GB / Monat (bei begründetem Antrag zeitlich begrenzt mehr).

- **Beschränkungen auf Deinen Speicherplatz**

Der Dir zugewiesene Plattenspeicher ist auf 5 GByte pro User beschränkt. Hast Du diesen Wert überschritten, hast Du keine weitere Speichermöglichkeit für Daten. **Insbesondere heißt das auch, dass Du keine Mail mehr bekommst.**

- **Benutzer-Account**

Die Weitergabe des Benutzer-Accounts an Dritte ist nicht gestattet.

- **Fenster**

Beim Verlassen der CIP-Pools bitte unbedingt alle Fenster schließen, da sonst vorallem im Sommer die Rechner überhitzen und sich der Raum in eine Sauna verwandelt.

Wiederholter Verstoß gegen die Benutzerordnung wird mit Entzug der Rechnererlaubnis bestraft. Je nach Schwere des Deliktes kann sogar eine Exmatrikulation des Täters erfolgen.

2 Einleitung

2.1 Woher stammt Linux?

Linux (Linus' Unix) ist ein Betriebssystem, das von Unix-Systemen abstammt. Es wurde im Jahre 1991 von dem finnischen Informatik-Studenten Linus Torvalds entwickelt; siehe dazu auch sein empfehlenswertes Buch „Just for Fun - Wie ein Freak die Computerwelt revolutionierte“. Sein Ziel war es, ein Unix-System, wie er es von den Großrechenanlagen der Universität kannte, auch auf seinem heimischen PC (einem i386) zu betreiben. Im Laufe eines Jahres ist aus dieser Idee ein selbst geschriebenes Betriebssystem entstanden, welches Linus ins Internet stellte, um es von Anderen testen zu lassen. Daraufhin beteiligten sich immer mehr Programmierer auf der ganzen Welt an der Weiterentwicklung dieses Systems und der Portierung von Anwendungen aus der Unix-Welt auf Linux.

2.2 Was ist Linux?

Linux an sich ist eigentlich nur der Betriebssystemkern (der Kernel), der die Hardware für die einzelnen Anwendungen abstrahiert. Der Kernel ist unter anderem zuständig für das Speichermanagement, das Multitasking und für den Umgang mit Geräten, den so genannten Devices.

Dieser Kern ist an sich nutzlos, da er alleine nicht in der Lage ist, für den Benutzer sinnvolle Dinge zu tun. Damit überhaupt sinnvolles Arbeiten möglich ist, gehören zum Betriebssystem eine Reihe von Systemprogrammen. Ein wichtiges, die bash, ein Kommandozeileninterpreter, wird später vorgestellt.

Alle anderen Programme sind die eigentlichen Anwendungen, die für bestimmte Aufgaben geschrieben worden sind, aber ohne die das System natürlich problemlos funktioniert.

Eine Linux-Distribution ist eine Programmsammlung, die neben dem Kernel und den Systemprogrammen auch eine Vielzahl von Anwendungsprogrammen beinhaltet. Distributionen sind getestete und aufeinander abgestimmte Programmpakete, die auch das eigentliche Betriebssystem enthalten. Zu den bekanntesten und verbreitetsten zählen unter anderem Ubuntu, Linux Mint, Fedora, OpenSUSE und Debian. Wir brennen regelmäßig neue Distributionen auf CD oder DVD und legen diese vor unserem Büro auf dem Aktenschrank aus.

2.3 Warum Linux?

- **Stabilität**

Linux ist ein sehr stabiles System. Selbst fehlerhafte Anwendungen bringen Linux nicht aus dem Tritt. Hardwaretreiber und neue Betriebssystemfunktionen werden erst nach vielen und ausführlichen Tests als stabil freigegeben.

- **Frei verfügbarer Sourcecode**

Die Quellen des Betriebssystems und der meisten Anwendungen liegen offen. Daher kann jeder Nutzer Fehler beheben, Sicherheitslücken schließen und Software auf Hintertüren überprüfen. Jeder kann und darf diese Software weiterentwickeln oder daraus neue Software entwickeln.

- **Sicherheit**

Ein System, welches die Daten der Benutzer vor anderen schützt, schützt auch die eigenen. Wegen der frei verfügbaren Quellen kann jeder Sicherheitslücken schließen, und Patches sind schneller zu haben. Die Stabilität trägt logischerweise auch zur Sicherheit bei.

- **Netzwerkbetriebssystem**

Der Netzwerkbetrieb ist von Anfang an Teil des Betriebssystems. Es ist möglich, Linux (wie auch alle anderen Unix-Systeme) komplett über das Netzwerk fernzusteuern.

- **Einfache Administration**

Standardschritte lassen sich mithilfe von Skripten leicht automatisieren; die Administration kann zentral erfolgen. Dies ist auch von jedem Rechner im Netzwerk möglich, also auch über Internet von zu Hause...

- **Multiuser-, Multitasking-, Multisession-Betriebssystem**

Linux unterscheidet Benutzer (Multiuser) und jeder Benutzer kann gleichzeitig mit mehreren Anwendungen arbeiten (Multitasking). Mehr noch, es können auch mehrere Benutzer an einem System gleichzeitig arbeiten oder ein Benutzer kann gleichzeitig mehrere Sitzungen an einem System geöffnet haben (Multisession).

Natürlich hat diese Liste keinen Anspruch auf Vollständigkeit. Auch die einzelnen Punkte sind knapp und unvollständig. Aber schließlich zählt nur, was jeder für sich als seine Vorteile für Linux entdeckt!



3 Einführung in Linux/Unix

3.1 Der Login

Unix-Systeme unterscheiden zwischen verschiedenen Benutzern. Deshalb muss man sich zuerst mittels einer Login-/Passwort-Kombination am System anmelden um einen Rechner zu verwenden. Der Login, oder besser gesagt der Benutzername im System, wird vom Systemadministrator vergeben und wird in der Regel aus dem Nachnamen des Benutzers erzeugt. Das Passwort wird zufällig generiert und muss nach Erhalt eines Accounts geändert werden (siehe ??). Der Login kann graphisch (siehe Abbildung ??) oder textbasiert erfolgen:

```
infcp10 login: goergen
Password:
Willkommen im CIP-Pool der Abteilung Informatik.
goergen@infcp10 _
```

Nach dem Anmelden hat man sich gegenüber dem System identifiziert. Alles was man nun macht wird vom System registriert und vieles, vor allem Unregelmäßigkeiten, werden vom System mitgeschrieben (geloggt). Falls jemand also seinen Account missbraucht und z. B. Angriffe auf andere Systeme ausübt, kann er durch diese Logs überführt werden!

Auf den Rechnern hat man die Wahl zwischen 6 Konsolen (nur Texteingabe) und der grafischen Oberfläche. Zwischen den Konsolen kann man mit **<Strg+Alt+F1>** bis **<Strg+Alt+F6>** hin- und herschalten. Zur grafischen Oberfläche bzw. dem grafischen Loginmanager, sofern dieser läuft, kommt man mit **<Strg+Alt+F7>**.

3.1.1 Passwort ändern

Unbedingt nach dem ersten Login das Passwort ändern! Dies geschieht in der Shell (siehe ??) mit dem Kommando `passwd`:

```
goergen@infcp70> passwd
Password:
Geben Sie ein neues UNIX Passwort ein:
```

Passwörter können leicht durch sogenannte „Brute force“-Methoden erraten werden. Das bedeutet, man probiert nacheinander alle Wörter aus, die in einem Wörterbuch gespeichert sind. Solche Wörterbücher existieren für alle Sprachen, aber auch als Spezialwörterbücher für Namen, Hobbies, Fachgebiete und vieles mehr.

Auch die vom Systemadministrator vergebenen Passwörter sind in der Regel zu unsicher, weil sie nach einem bestimmten Verfahren generiert wurden. Da sich in einem Unix-System zurückverfolgen lässt, wer wann wo was gemacht hat, kann jemand, der die Login-/Passwort-Kombination eines Anderen besitzt, auf dessen Kosten Unfug treiben und nicht erwischt werden. Schuld ist der Besitzer dieses Logins!



Abbildung 1: Ein graphischer Login.

3.1.2 Passwörter sicher wählen

Sichere Passwörter enthalten:

- Sonderzeichen und Zahlen
- Groß- und Kleinschreibung (wird unterschieden!)
- Keine ganzen Wörter
- und sind mindestens 8 Zeichen lang

Um sich solche kryptische Passwörter zu merken gibt es folgende Möglichkeit: Man überlegt sich einen Satz (z. B. ein Zitat, ein Liedtext,...) und benutzt die Anfangsbuchstaben (Endbuchstaben,...) und die Satzzeichen als Passwort: **Ich studiere Informatik im 1. Semester! = IsIi1.S!**

Falls keine Satzzeichen vorkommen, fügt man an bestimmten Stellen Sonderzeichen ein oder ersetzt damit andere Zeichen zum Beispiel statt „a“ ein @ oder statt „s“ ein \$ oder statt „und“ ein & oder +, usw.

3.2 Bedienungsoberfläche

Die graphische Benutzeroberfläche vereinfacht das Starten der wichtigsten Anwendungen und bietet die Möglichkeit, einfacher mit mehreren Anwendungen parallel zu arbeiten. In Unix-Systemen wird sie X-Window genannt. Ihr Aussehen und Verhalten wird von der Desktop-Umgebung bestimmt. Bekannte Desktop-Umgebungen sind GNOME, KDE, LXDE oder auch Xfce. Letztere wird wegen ihrer Schnelligkeit in den CIP-Pools eingesetzt.

Xfce hat ein ähnliches „Look and Feel“ wie Windows. In der Standard-Konfiguration ist links unten ein Startmenü zu finden, aus dem sich ausgewählte Anwendungen starten lassen. Unten befindet sich die auch von Windows bekannte Taskleiste, die alle aktuell laufenden Programme anzeigt, und schnell anwählbar macht. Daneben liegt eine Schnellstartleiste mit wichtigen Programmen und weiteren Katalogen. Rechts in der Ecke befinden sich neben der Uhr auch Symbole laufender Dienste. Diese sind aber bei weitem nicht vollständig. Im Hintergrund liegen auf dem Desktop die Icons einiger Programme, die damit schnell gestartet werden können.

Im Unterschied zu Windows kennen die meisten X-Window-Oberflächen mehrere Desktops. Auf diesen können sich unterschiedliche Programme befinden. Jeder Desktop besitzt seine eigene Taskleiste. So kann man laufende Programme nach Zweck sortieren (Browser, Mailclient, Pidgin, Office-Dokumente, ...) und schnell zwischen den einzelnen wechseln. Dieses Wechseln geschieht durch die Tastenkombination **<Strg+Alt+Pfeil (links oder rechts)>**, durch Klicken auf die Desktopsymbole neben der Taskleiste oder, wenn eingerichtet durch Bewegen der Maus über den Bildschirmrand hinaus.

3.3 Die Shell (Kommandozeile, Terminal, Konsole)

Das wichtigste Programm in Unix Systemen ist die Kommandozeile, die auch die Shell oder Terminal genannt wird. Die Shell ist eine „Schale“ über dem Betriebssystemkern (Kernel) und bietet eine benutzerfreundliche Abstraktion des Betriebssystems. Mit einer Shell besitzt man Zugang zu allen Funktionen des Betriebssystems, wenn man die Rechte dazu hat siehe hierzu Abschnitt ???. Eine Shell ist ein sogenannter Kommandozeileninterpreter; das bedeutet, sie wartet auf Benutzereingaben, führt eingegebene Kommandos aus und kann dann wieder weitere Eingaben entgegennehmen. Die Eingabebereitschaft zeigt die Konsole durch eine Eingabeaufforderung an, den sogenannten Prompt:

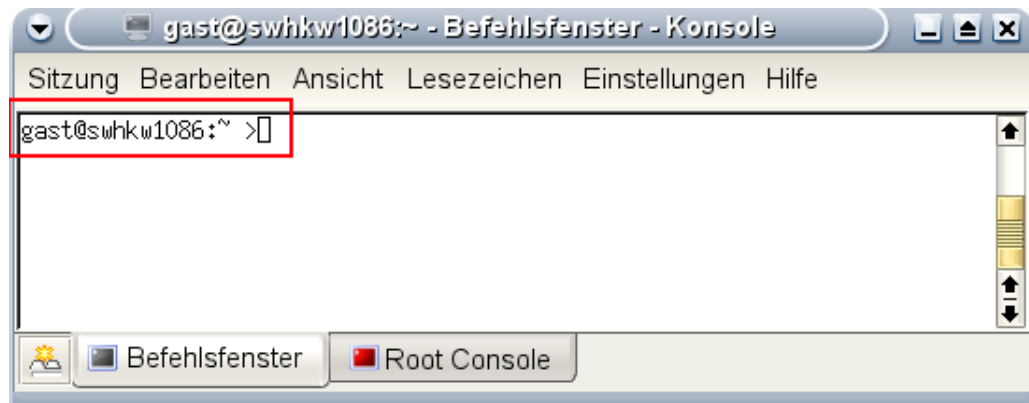


Abbildung 2: Konsole mit Prompt in einem Fenster unter KDE.

Hinter diesem Prompt können Befehle eingegeben werden. Der Prompt kann je nach Konfiguration unterschiedlich aussehen. Im obigen Beispiel enthält der Prompt den Loginnamen *gast* des Benutzers. Das @ ist das Trennzeichen zwischen dem Loginnamen und dem Rechner, auf dem der Benutzer eingeloggt ist. Im obigen Beispiel hat der Rechner den Namen *swkw1086*. Am Ende des Prompts steht ein Zeichen nach dem die Eingabe erfolgt, in unserem Fall ist es „>“, es kann auch „#“ oder „\$“ oder etwas anderes sein. „|“ bezeichnet im Beispiel den Cursor, den aktuellen Punkt der Eingabe.

Auf der Konsole können nicht nur Kommandos an das Betriebssystem und die Shell übergeben werden z. B. „Ändere das aktuelle Arbeitsverzeichnis der Shell in /usr/share“: `cd /usr/share`. Sondern es können auch „große“ Anwendungen wie z. B. `firefox` gestartet werden.

3.3.1 Warum die Shell verwenden die graphische Oberfläche ist doch viel komfortabler?

Komfortabler ist sie in vielen Fällen, aber nicht in allen! Denn eine graphische Oberfläche macht eigentlich nichts anderes als Benutzereingaben auf eine Shell abzubilden. Daher hat sie fast immer eine geringere Mächtigkeit als eine Konsole. In anderen Worten: mit einer Shell lässt sich mehr anstellen als mit der Oberfläche. Viele Aufgaben sind mit der Shell schneller erledigt und häufig wiederkehrende Aufgaben lassen sich dort durch Skripte sehr leicht automatisieren. Eine Shell kann auch über ein Netzwerk in vollem Umfang bedient werden.

3.3.2 Kommandos

Kommandos haben immer folgenden Aufbau:

`Kommandoname [-Option(en)] [Argument(e)]`

Am Anfang ein Kommandoname, eventuell gefolgt von einer oder mehreren Optionen und einem oder mehreren Argumenten. Zwischen dem Kommando, den Optionen und den Argumenten stehen **Leerzeichen**. Befinden sich drei Punkte (...) hinter den eckigen Klammern dann kann man mehrere Optionen/Argumente angeben. Die rechteckige Klammern „[]“ bedeuten, dass etwas optional angegeben werden kann. Als Beispiel nehmen wir das Kommando `ls`, Abkürzung für das englische „list“, dieser listet den Inhalt eines Verzeichnisses auf:

```
$ ls [OPTION]... [FILE]...
$ ls -al /home/jacob
```

Nach dem Befehl **ls** stehen zwei Optionen, **a** und **l**. Durch die Option **a** werden auch die versteckten Dateien aufgelistet und durch die Option **l** werden zusätzliche Informationen (Benutzerrechte, Erstellungsdatum, ...) der Dateien angezeigt. Nach den Optionen wurde das Verzeichnis `/home/jacob` angegeben, somit stellt **ls** den Inhalt dieses Verzeichnisses dar. Wird kein Ordner angegeben dann wird das aktuelle Verzeichnis aufgelistet und da sich hinter `[FILE]` drei Punkte befinden, hätte man auch mehrere Verzeichnisse angeben können.

3.4 Das Dateisystem

Ein Dateisystem ist eine Ordnungsstruktur für Daten. Daten und Programme werden in einzelnen Dateien abgelegt. Damit man nicht die Übersicht verliert, sortiert man diese Dateien in Ordnern (so genannten Verzeichnissen) und diese wieder in Ordnern, so dass man Daten katalogisieren kann. (Vergleiche Leitz-Ordner mit weiteren Einhängeordnern, die wiederum Blätter enthalten. Diese stehen in Regalen, Räumen,...) Dadurch entsteht eine hierarchische Ordnung von Dateien und Verzeichnissen, die sich durch einen umgekehrten Baum darstellen lässt. Dieser Baum hat eine eindeutige **Wurzel** oder **Root**, die keine übergeordneten Ordner besitzt.

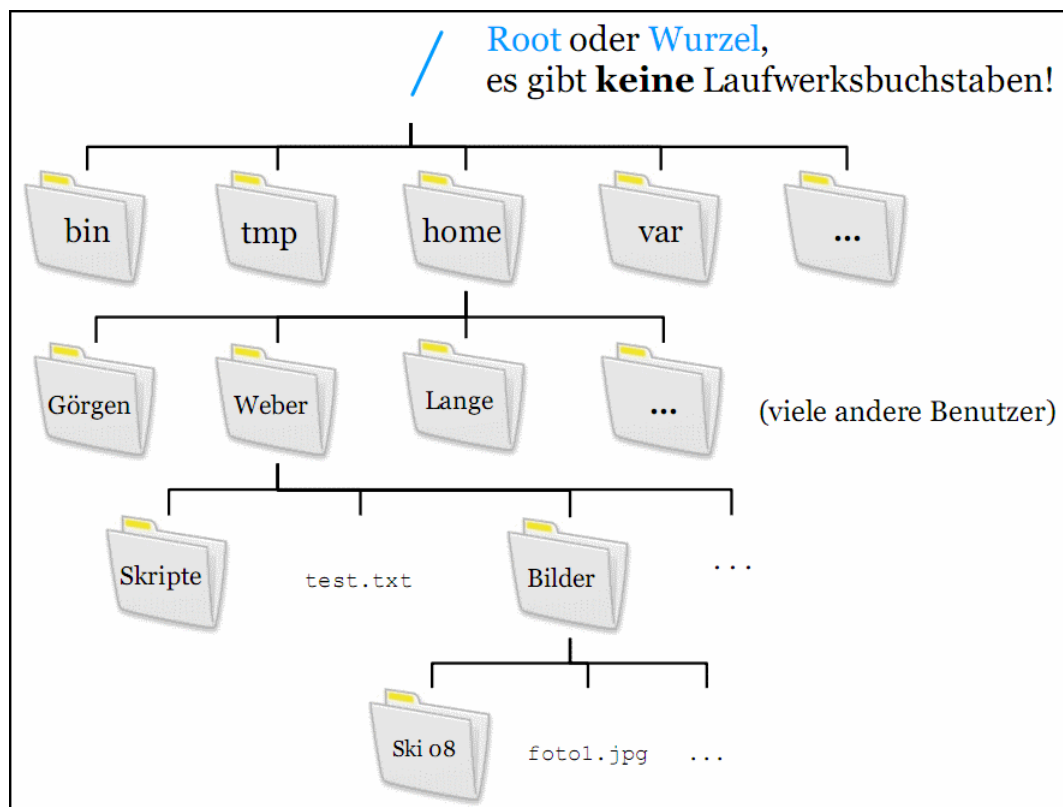


Abbildung 3: Ein Verzeichnisbaum. Grau unterlegt und abschließendes `/` bedeutet, dass es sich um ein Verzeichnis handelt. Alles andere sind Dateien. Im Unterschied zu Windows gibt es unter Linux keine Laufwerksbuchstaben.

3.4.1 Navigieren im Verzeichnisbaum

Das Arbeitsverzeichnis ist das Verzeichnis, in dem man sich aktuell befindet. Programme in diesem Verzeichnis können direkt gestartet werden und Dateien direkt (ohne Pfadangabe) geöffnet werden. Der Befehl **pwd** (**p**rint **w**orking **d**irectory) gibt einem das aktuelle Arbeitsverzeichnis an. Mit **ls** wird der Inhalt eines Verzeichnisses angezeigt ($\rightarrow$?? Kommandos). Um in ein Verzeichnis zu wechseln benutzt man den Befehl **cd** *Verzeichnis* (**c**hange **d**irectory).

`$ cd /usr/share/doc` ➤ Mit **cd** wechseln wir ins `doc` Verzeichnis ($\rightarrow$?? Dokumentation).

Verzeichnis- und Dateipfade kann man absolut oder relativ angeben. Absolute Angaben macht man von der Verzeichniswurzel aus und nennt alle Verzeichnisse, die auf dem Weg zum Zielverzeichnis liegen z. B. `/usr/bin/firefox`. Relative Angaben beziehen sich immer auf das aktuelle Arbeitsverzeichnis und beinhalten alle Verzeichnisse, die auf dem Weg vom aktuellen Verzeichnis zum Zielverzeichnis liegen. Dabei bezeichnet das spezielle Verzeichnis „`..`“ das übergeordnete Verzeichnis.

Nehmen wir an, das aktuelle Arbeitsverzeichnis sei `/home/Weber` aus dem Verzeichnisbaum in Abbildung ?? . Dann ergeben folgende absolute und relative Angaben einen Sinn:

absolut: <code>/home/Weber/Bilder/foto1.jpg</code>	→ relativ: <code>Bilder/foto1.jpg</code>
absolut: <code>/home/Lange/</code>	→ relativ: <code>../Lange</code>
absolut: <code>/usr/bin/firefox</code>	→ relativ: <code>../../usr/bin/firefox</code>

Sehr viele Konsolenbefehle benötigen Pfadangaben, und vielen Programmen kann man so Dateinamen angeben, die sie öffnen sollen z. B.:

```
$ firefox /home/goergen/html/index.html
```

Das Home-Verzeichnis ist das Verzeichnis, in dem der Benutzer seine privaten Daten ablegen darf und zu dem andere Benutzer in der Regel keinen Zugriff haben. Es heißt meistens `/home/loginname` z. B. `/home/goergen`. Verwendet man `[cd ohne Pfad]` oder `[cd ~]`, so wechselt man in sein Home-Verzeichnis. Das aktuelle Arbeitsverzeichnis wird durch einen Punkt `[.]` bezeichnet. Beispiele:

```
$ cd .           Das Arbeitsverzeichnis ändert sich nicht!
$ cd ../../..    wechselt drei Ebenen nach oben
$ cd ~           wechselt in das Home-Verzeichnis
$ cd ~Loginname_deines_CIP-Pool_Nachbarn
```

Versteckte Dateien und Verzeichnisse lassen sich an dem Punkt vor ihrem eigentlichen Namen erkennen zum Beispiel `.config` oder `.bashrc`.

3.5 Arbeiten mit Dateien und Verzeichnissen

3.5.1 Verzeichnisse anlegen und löschen

Mit `mkdir Verzeichnis`, abgeleitet von **make directory**, erstellt man ein oder mehrere Ordner.

```
$ mkdir Fotosammlung Mathematik - Erstellt zwei Verzeichnisse.
```

Mit `rmdir Verzeichnis`, abgeleitet von **remove directory** oder `rm -r Verzeichnis` (`-r` steht für rekursiv) löscht man ein Unterverzeichnis.

```
$ rmdir Fotosammlung - Löscht das Verzeichnis Fotosammlung.
$ rm -r Mathematik - Löscht das Verzeichnis Mathematik.
```

3.5.2 Dateien lesen, erstellen, kopieren und löschen

Mit dem Programm `less` kann man große Textmengen lesen. Durch Drücken der Aufwärts- und Abwärtspfeile auf der Tastatur bewegt man sich im Text rauf und runter. Man verlässt `less`, indem man die Taste `Q` drückt:

```
$ less .bashrc
```

Selbstverständlich gibt es auch unter Linux Texteditoren wie zum Beispiel nano, vi, kate, oder gedit. Doch es ist auch möglich Textdateien ohne Editor zu erstellen:

```
$ echo "Bald ist die nächste Lanparty" > blub.txt
```

Der Pfeil „>“ leitet den Text in die Datei blub.txt weiter (→ ?? Ein- und Ausgabeumleitung).

Mit dem Befehl `cp Datei1 Datei2`, abgeleitet von **copy**, wird eine Datei oder ein Verzeichnis kopiert. Dabei sollte man vorsichtig sein, da cp eine vorhandene Datei ohne Nachfrage überschreibt. Im nachfolgendem Beispiel wird die *crashkurs.txt* Datei im Verzeichnis `/home/fsrinfo/WWW`, ins aktuelle Arbeitsverzeichnis (dargestellt durch den Punkt!) kopiert:

```
$ cp /home/fsrinfo/www/crashkurs.txt .  
$ cp crashkurs.txt robin.txt - Kopiert die Datei crashkurs.txt.
```

Durch die Eingabe von `rm Dateiname`, abgeleitet von **remove**, löscht man eine Datei.

```
$ rm robin.txt - Löscht die Datei robin.txt.
```

3.5.3 Verzeichnisse und Dateien verschieben

Um eine Datei oder ein Verzeichnis zu **verschieben** empfiehlt sich der Befehl `mv Datei1 Datei2`, abgeleitet von **move**. Dieser kann außerdem zum **Umbenennen** von Dateien benutzt werden.

```
$ mv blub.txt lan.txt - Benennt die Datei blub.txt in lan.txt um.
```

3.5.4 Verzeichnisse oder Dateien verknüpfen

Der Befehl `ln VorhandeneDatei Zielname`, abgeleitet von **Link**, verknüpft eine Datei oder ein Unterverzeichnis mit einem Namen.

Im ersten Beispiel erstellen wir eine Verknüpfung namens **www** im Homeordner auf das Webverzeichnis `/pub/www/loginname`.

```
$ cd > ln -s /pub/www/loginname www > ls -l >
```

Im zweiten Beispiel erzeugen wir einen Link namens **wurzel** im Homeordner auf das Rootverzeichnis und löschen diesen am Ende:

```
$ cd > ln -s / wurzel > cd wurzel > cd > rm wurzel >
```

3.5.5 Den belegten Speicherplatz ermitteln

`du Verzeichnis`, abgeleitet von **disk usage**, zeigt alle Dateien und deren Größe an, die im angegebenen Verzeichnis existieren.

```
$ cd > du Desktop > du -h . > du -hs . >
```

3.5.6 Durchsuchen mit Grep

`grep Zeichenfolge Datei`, abgeleitet von **global/regular expression/print**, durchsucht Dateien nach Zeichenfolgen.

```
$ grep wann lan.txt  
Grep durchsucht die Datei lan.txt nach Zeilen mit dem Wort wann und gibt diese aus.
```

3.5.7 Finden mit Find

`find` *Zielordner -Optionen Argument* durchsucht Verzeichnisse und zwar rekursiv das bedeutet alle Unterordner des durchsuchten Ordners werden auch durchsucht. Man kann `find` mit regulären Ausdrücken kombinieren, siehe ?? Einfache reguläre Ausdrücke. Achtet darauf bei regulären Ausdrücken Klammern zu setzen wie im Beispiel, sonst wandelt die Shell zuerst den regulären Ausdruck um und `find` wird dann nach einem anderen Begriff suchen als von euch erwartet.

```
$ find . -name "lan.txt" > find . -name "?an.*" >
```

Sucht zuerst ohne dann mit regulären Ausdrücken nach der *lan.txt* im aktuellen Ordner.

3.5.8 Komprimierte und archivierte Dateien

Im Laufe Deines Studiums wirst Du häufig mit Dateien zu tun haben, die folgende Endungen haben:

- **.gz** und **.zip** Diese Dateien sind komprimiert. Der Befehl `gunzip skript.gz` dekomprimiert die Datei *skript.gz* danach wird diese Datei zu der Datei *skript*. Mittels `unzip irgendwas.zip` lässt sich die Datei *irgendwas.zip* dekomprimieren.

- **.tar** Diese Dateien beinhalten Archive, d.h. mehrere Dateien und Verzeichnisse, die in einer Datei zusammengefasst sind.

Der Befehl `tar -xf irgendwas.tar` entpackt das Archiv *irgendwas.tar* in das aktuelle Verzeichnis. Die Datei *irgendwas.tar* wird nicht gelöscht.

`tar -xf irgendwas.tar -C desktop` entpackt das Archiv *irgendwas.tar* in das Verzeichnis *desktop*.

- **.tar.gz** oder **.tgz** sind komprimierte Archive.

`tar -xzf irgendwas.tar.gz` entpackt das Archiv *irgendwas.tar.gz* in das aktuelle Verzeichnis. Die Datei wird nicht gelöscht.

`tar -xzf texte.tar.gz -C docs` entpackt das Archiv *texte.tar.gz* in das Verzeichnis *docs*. Die Datei wird nicht gelöscht. Dasselbe gilt für **.tgz** Dateien.

Falls Du selber Dateien komprimieren oder archivieren willst:

- `gzip datei`
Komprimiert die Datei *datei*, das Ergebnis ist die Datei *datei.gz*
- `gzip -c datei > irgendwas.gz` Komprimiert die Datei *datei*, das Ergebnis ist Datei *irgendwas.gz*, die Datei *datei* bleibt unverändert.
- `tar -cf irgendwas.tar urlaub` Erzeugt ein Archiv namens *irgendwas.tar* aus allen Dateien und Verzeichnissen, die sich im Verzeichnis *urlaub* befinden.
- `tar -czf irgendwas.tgz party` Erzeugt ein komprimiertes Archiv namens *irgendwas.tgz* aus allen Dateien und Verzeichnissen, die sich im Verzeichnis *party* befinden.
- `zip irgendwas.zip datei1 datei2 ...` Erzeugt ein komprimiertes Archiv namens *irgendwas.zip* aus allen angegebenen Dateien; man kann auch Verzeichnisnamen angeben.

3.6 Das Leben mit dem Terminal

Mit einigen Tricks lässt sich deutlich schneller mit dem Terminal arbeiten. Wenn man sich nun vorstellt, dass es Verzeichnispfade auf Unix-Systemen gibt die zum Beispiel `/usr/X11R6/lib/X11/fonts` oder `/usr/local/man/man1/en/man.1.gz` heißen, dann ist es unangenehm, wenn man diese vollständig tippen muss. Deshalb existieren diverse Hilfsmittel, um sich das Schreiben zu erleichtern.

- **Tab-Expansion** Drückt man in einem angefangenen Ausdruck die Tab-Taste, dann wird die Shell versuchen diesen Ausdruck zu expandieren. Befindet man sich z. B. im Homeordner und man drückt nach `ls De` die Tabtaste, vervollständigt die bash den Ausdruck als `ls Desktop`, sofern keine weitere Datei mit `De` beginnt. Gibt es mehrere Möglichkeiten dann können diese durch 2x Tabtastendruck angezeigt werden.
- **Command History** Durch Drücken der Aufwärts und Abwärtspeile auf der Tastatur kannst du die schon eingegebenen Kommandos wiederholen. Um in der History nach vorherigen Befehlen zu suchen, drückt man `<Strg+R>`. In der Regel werden bis zu 500 Befehle in der Datei `~/.bash_history` gespeichert. Es ist möglich die Größe der History selbst festzulegen. Und zwar indem man `export HISTFILESIZE=5000` in der erste Zeile der `.bashrc` Datei (liegt im Home Ordner) hinzufügt und so den Wert auf z. B. 5000 Einträge erhöht.
- **Editieren in der Kommandozeile** Man kann in der Kommandozeile editieren, will heißen Zeichen durch andere ersetzen, hinzufügen und entfernen (anders als bei DOS). Dies geschieht wie in einem „normalen“ Editor.
- **Kopieren mit der Maus** Unter Linux gibt es die die 3-Tasten-Maus. Kopieren geht folgendermaßen: Mit der linken Maustaste einen Textbereich auswählen, dann die Maus an eine andere Stelle bewegen (auch in einem anderen Fenster, Programm...) und die mittlere Maustaste drücken - klappt fast immer.
- **Aliase in .bashrc** In der `.bashrc` Datei, die in deinem Home Ordner liegt, kannst du Aliase festlegen. Willst du z. B., dass `ls` immer mit der Option `--color` aufgerufen wird, schreibst du in die `.bashrc` Datei: `alias ls="ls --color"`. Wenn du dir nicht sicher bist ob ein Befehlsname schon vergeben ist, kannst du dies im Terminal mit `type Befehlsname` nachprüfen.

3.7 Datei- und Benutzerrechte

Linux ist ein Mehrbenutzersystem. Das bedeutet, dass theoretisch jeder Benutzer auf die Daten der anderen Benutzer zugreifen könnte. Aus verschiedenen Gründen sollte aber gerade dies vermieden werden. Deshalb kann der Eigentümer einer Datei entscheiden, wer auf seine Datei zugreifen darf. Wie man herausfindet, wer Zugriff auf bestimmte Dateien hat und wie man diese Rechte ändert, wird im Folgenden erklärt.

Genauere Informationen über Dateien und Verzeichnisse erhält man mit dem Befehl `ls -l` (long):

```
goergen@infcp10> ls -l
-rw-r--r---  1    1286  manager1  staff    bild.gif
-rwxr-x---   1   22078  manager1  manager  programm
-rw-rw-r--   1   5319   manager1  manager  out.txt
drwxr-xr-x   1   1024   manager1  staff    skripte
Typ und Rechte  Links  Größe  Eigentümer  Gruppe  Dateiname
```

In der ersten Spalte befinden sich jeweils immer 10 Zeichen. Das erste Zeichen bestimmt den Typ der Datei. Verzeichnisse beginnen mit einem **d** wie **directory** (z. B. **drwxrwxrwx**). Dateien mit Daten wie Textdateien beginnen mit einem **-** (also **-rw-rw-rw-**). Links beginnen mit einem **l** und besondere Dateien für Geräte mit **c** oder **b**.

Die 9 weiteren Zeichen (**rwX rwX rwX**) sind in 3 Blöcke aufgeteilt. Sie definieren die Zugriffsrechte des Eigentümers (oder auch **owner**, auch oft als **user** bezeichnet), der Gruppe (**group**) und der Anderen (**others**) auf eine Datei. **RWX** steht übrigens für **read**, **write** und **execute**. Hat eine Datei die Rechte **rw-rw-r--** dann kann jeder die Datei lesen (**read**) aber nur der Eigentümer und die Gruppe können die Datei modifizieren (**write**).

Die Bedeutung von **rwX-** für Dateien und Verzeichnisse ist in der folgenden Tabelle dargestellt:

Rechte	Dateien	Verzeichnisse
read	Leserecht - lesen aber nicht verändern	Recht, den Inhalt eines Verzeichnisses aufzulisten.
write	Schreibrecht - bearbeiten und löschen möglich	Recht, im Verzeichnis Dateien zu erzeugen und zu verändern wenn Dateirechte es erlauben.
execute	Recht, diese Datei auszuführen.	Recht, ins Verzeichnis zu wechseln
-	Dieses Recht ist nicht vergeben.	

Dateien haben immer einen eindeutigen Eigentümer. Darüber hinaus gehören Dateien zu einer Gruppe. Alle Benutzer (oder weitere Gruppen), die zu dieser Gruppe gehören, besitzen die Gruppenrechte. Jeder Benutzer kann zu einer oder mehreren Gruppen gehören.

Die folgenden Beispiele sind zum besseren Verständnis der Benutzer- und Dateirechte gedacht:

typ	owner	group	others	Erklärungen:
d	rwX	r-X	---	Group darf ins Verzeichnis wechseln, aber nur lesen, owner darf alles, die Anderen nichts.
d	rwX	--X	---	Group darf zwar ins Verzeichnis wechseln, aber noch nicht einmal die Dateien listen.
-	rwX	r-X	r-X	Ausführbare Datei, die nur vom owner geändert werden darf.
-	rw-	r--	---	Einfache Datei, owner darf lesen und schreiben, die Gruppe nur lesen, der Rest nichts.

Mit **chmod** (**chmod [User] [+ oder -] [rwX]_[Dateiname]**) kann der Eigentümer einer Datei alle Zugriffsrechte verändern. [User] kann **u**(user), **g**(group), **o**(others) oder **a**(all) sein. Mit dem **+** vergibt man Rechte und mit dem **-** entzieht man einem Benutzer Rechte. Beispiele:

```
$ chmod a+rw liste.txt  gibt jedem (a) das Recht die Datei liste.txt zu lesen (r) und zu
                        verändern (w)
$ chmod u-x skript.txt  entzieht dem user (u) das Recht die Datei ausführen (-x)
$ chmod ug+rw math.txt  user und group erhalten die Rechte read und write
```

Alternativ kann man die Form **chmod [E G A]_[Dateiname]** verwenden. **E** steht für die Rechte des Eigentümers, **G** der Gruppe, **A** der Anderen. **E**, **G**, **A** sind jeweils Zahlen von 0-7. Aufgepasst nicht mit 0-700 verwechseln, falsch wäre z. B.: 128 oder 599. Jede Zahl (**E**, **G**, **A**) steht für ein Benutzerrecht: **0 = - - -** **1 = - - x** **2 = -w-** **3 = -wx** **4 = r - -** **5 = r-x** **6 = rw-** **7 = rwX**
Beispiele:

```
$ chmod 664 blub.txt      → user und group bekommen read und write, others nur read
$ chmod 777 passwort.txt  → jeder bekommt read, write und execute Rechte
```

3.8 Hilfe

Hilfe erhält man unter Linux auf vielen Wegen. Bevor man Andere mit häufig gestellten Fragen nervt (FAQ = Frequently Asked Questions) und deren Antworten einfach nachzulesen sind, sollte man folgende Quellen berücksichtigen. Der Großteil der Dokumentation und der Hilfeseiten ist in englischer Sprache. Dies sollte jedoch keinen abschrecken.

3.8.1 Manpages und die wichtigsten Befehle (**man**, **whatis**, **apropos**, **-h**, **info**)

Der Anlaufpunkt, wenn man wissen will, wie ein Befehl zu verwenden ist, sind die „Manual Pages“. Hier sind nahezu alle Kommandos und Programme zu finden. Die Man Page zu einem Befehl erhält man durch die Eingabe von **man *Befehl*** (z. B.: **man pwd**). Scrollen kann man mit den Cursortasten und beendet wird man mit der Taste **Q**. Mit **/** und anschließendem Suchbegriff kann man die Manpages durchsuchen. Wer mit **man** nicht zurecht kommt, sollte **man man** ausprobieren ;-).

Eine Kurzbeschreibung eines Befehls erhält man mit **whatis *Befehl*** (z. B.: **whatis cp**).

Mit **apropos *pattern*** lassen sich alle Kurzbeschreibungen anzeigen, die ein bestimmtes Muster (engl. pattern) enthalten. Das Beispiel **apropos list** gibt alle Beschreibungen aus, die das Muster *list* beinhalten, unter anderem auch die Beschreibung von **ls**.

Bei sehr vielen Befehlen führen die Optionen **--help** oder **-h** (z. B.: **ls --help**) zur Ausgabe einer kurzen Befehlsreferenz.

Die Info-Seiten bieten für viele Programme eine Bedienungsanleitung (während die Manpages meist nur die Optionen und Argumente des Aufrufs beschreiben und das, was das Programm macht). So kann man erfahren wie ein Programm zu verwenden ist. Existieren keine Info-Seiten, zeigt **info *Befehl*** (z. B.: **info ls**) die eventuell vorhandene Man-Page an.

3.8.2 Weitere Hilfe in der Dokumentation und im Internet

Darüber hinaus finden sich im Verzeichnis **usr/share/doc** häufig Informationen und zusätzliche Dokumentation zu einzelnen Programmpaketen. Hier sind auch (Online-)Handbücher zu finden.

Wenn die oben genannten Hilfsmittel einen nicht zum Ziel bringen und auch der Nachbar im Pool nicht weiterhelfen kann (oder man gerade nicht im Pool sitzt...), gibt es noch die Möglichkeit, auf verschiedenen Seiten im Internet Hilfe zu erlangen z. B. für Dokumentationen **linuxdoc.org**, **tldp.org**, auf den Seiten der Distributoren **debian.org**, **ubuntu.com**, in Foren **linuxquestions.org** oder auch in Linux-Newsgroups **comp.os.linux.***, **de.comp.os.unix.linux.*** ... wer sucht der findet!

3.8.3 Admins fragen!

Natürlich sind unsere Admins von der freundlichsten und hilfsbereitesten Sorte! Leider sind sie dilettantische Java-Programmierer, daher können sie euch dabei in keinsten Weise helfen. Sie besitzen übrigens weder Tacker noch Büroklammern. Bitte wendet euch diesbezüglich lieber an uns. ;)

3.9 Prozesse (Prozess = Befehl, Programm)

Linux ist ein Multitasking-Betriebssystem. Das bedeutet, dass mehrere Prozesse quasi gleichzeitig ablaufen. Prozesse, die nichts miteinander zu tun haben, beeinflussen sich nicht und haben ihren eigenen „privaten“ Bereich, in dem sie ungestört werkeln können. Jeder Prozess ist einem Benutzer zugeordnet und besitzt dessen Benutzer- und Gruppenrechte; er darf nur auf Dateien zugreifen, auf die der Benutzer auch zugreifen darf. In einem Standard-Linuxsystem laufen vom Start an schon sehr viele Prozesse. Diese gehören zu Betriebssystemdiensten und laufen im Hintergrund.

Mit dem Befehl **top** werden alle Prozesse dynamisch angezeigt. An erster Stelle steht der Prozess der am meisten Ressourcen beansprucht. Man erkennt an den Spalten CPU und MEM wieviele Ressourcen ein Prozess benötigt.

Der Befehl **ps** (**p**rocesses) zeigt laufende Prozesse an. **ps aux** (a = alle, u = Benutzernamen und PIDs anzeigen, x = extended) zeigt auch die Prozesse anderer Benutzer an und weitere zusätzliche Informationen.

Jeder Prozess hat eine eindeutige **PID** (Process Identification). Zu einem Programm können mehrere Prozesse gehören. Mit **pidof Programmname** (z. B. **pidof firefox**) lassen sich die PIDs eines Programmes anzeigen.

3.9.1 Hintergrundprozesse

Kommandos führen nur kurze Aktionen aus und ermöglichen es, die Shell weiter zu verwenden. Viele Programme (vor allem die mit graphischen Oberflächen) arbeiten länger, und „blockieren“ damit die Shell, d.h. erst nach Beenden des Programms erscheint wieder die Eingabeaufforderung. Will man die Shell aber weiterverwenden, kann man Programme mit einem nachgestellten **&** in den Hintergrund schicken (z. B.: **firefox &**). Die Shell ist dann sofort wieder eingabebereit.

Ist eine Shell schon blockiert, kann man das laufende Programm mit der Tastenkombination **<Strg+Z>** unterbrechen und mit dem Befehl **bg %Jobnummer** (**b**ackground) in den Hintergrund schicken. Mit **fg %Jobnummer** (**f**oreground) erscheint das Programm wieder im Vordergrund. Will man das Programm beenden, drückt man **<Strg+C>**, C für **C**ancel.

Der Befehl **jobs** zeigt alle Prozesse an, die in der aktuellen Shell gestartet wurden. Durch hinzufügen der Option **-l** lassen sich zusätzlich die PIDs angeben.

3.9.2 Signale

Prozessen kann man Signale senden, die Einfluss auf den Programmablauf haben können. Nur an eigene Prozesse kann man Signale schicken. Signale kann man mit dem Befehl **kill** senden. Die wichtigsten sind:

TERM	terminiert einen Prozess. Dieser darf sich korrekt beenden. z. B.: kill -TERM 4572 gleichbedeutend mit kill 4572
KILL	tötet einen Prozess. Dieser hat keine Möglichkeit sich korrekt zu beenden um z. B. Daten zu sichern. Beispiel: kill -KILL 1234 gleichbedeutend mit kill -9 1234
STOP	hält einen Prozess an.
CONT	führt einen angehaltenen Prozess fort.

Dem Befehl **killall** übergibt man den Namen des Programms. Das Signal wird allen Prozessen gesendet, die zu diesem Programm gehören: **killall -TERM nano** terminiert alle Prozesse die zu Nano gehören.

3.10 Ein- und Ausgabeumleitung

Viele Kommandos und Programme erzeugen Textausgaben auf der Shell. Diese Ausgaben können mit **>** in eine Datei gespeichert (umgeleitet) werden. Dabei muss man aufpassen, denn **>** überschreibt vorhandene Dateien.

\$ ls > listing - Das Verzeichnislisting wird in der Datei *listing* gespeichert.

>> hingegen hängt die Ausgabe an das Ende einer vorhandenen Datei an oder schreibt eine neue Datei, wenn noch keine existieren sollte. Zusätzlich lässt sich die Fehlerausgabe per **2>** umleiten (diese wird nicht mit **>** umgeleitet!).

Außerdem können viele Kommandos und Programme Benutzereingaben auf der Shell entgegennehmen. Diese Eingaben können auch einer Datei entnommen werden:

\$ grep gif < listing - Der Inhalt der Datei *listing* wird als Eingabe des Kommandos **grep** verwendet. **grep** sucht in der Eingabe nach Zeilen mit dem Muster „gif“ und gibt diese aus.

Die Ausgabe eines Programms kann aber auch direkt an die Standardeingabe eines anderen Programms umgeleitet werden. Die sogenannte Pipe (**|**) verbindet die beiden Programme:

\$ ls | grep gif - Listet alle Dateien im aktuellen Verzeichnis, die den Text „gif“ beinhalten.

Beispiele:

\$ ls -l | wc -l (**wc**: word count) zählt die Dateien im aktuellen Verzeichnis.

\$ ps -aux | grep walter listet alle Prozesse des Nutzers *walter* auf.

Die Eingabe- und Ausgabeumleitung ist sehr hilfreich, wenn man Dinge automatisieren will. Eingesetzt wird sie z. B., wenn Programme im Hintergrund laufen, aber immer noch Ausgaben auf der Shell abbilden sollten. Diese Ausgaben kann man entweder in eine Datei umleiten oder mit ***kommando*** **> /dev/null** ins Nirvana schicken.

3.11 Drucken auf der Shell

Zum Drucken von Dateien über die Shell verwendet man überwiegend **lp** oder **lpr**. Diese Befehle senden die Druckaufträge an den Standarddrucker. Mit der Option **-P** gefolgt vom Druckernamen wählt man einen alternativen Drucker aus. Zu beiden Befehlen gibt es noch einige Optionen die in den Manpages gut erklärt sind.

\$ lpr text.txt - Die Datei *text.txt* wird der Warteschlange des Standarddruckers angefügt.

\$ lpr -P A4_MINOLTA_COLOR text.txt

Die Datei steht nun in der Warteschlange des jeweiligen Druckers. Mit **lpq** lässt sich herausfinden welche deiner Dateien in der Warteschlange stehen. Auch hier lässt sich der gewünschte Drucker mit der Option **-P Druckername** auswählen, sonst erscheint die Warteschlange des Standarddruckers.

Sollte sich eine Datei nicht drucken lassen oder du willst sie aus der Warteschlange löschen damit sie nicht gedruckt wird dann solltest du **lprm** oder **cancel** benutzen. Beide Befehle verlangen nach dem Namen (eine Nummer) des Druckauftrages welcher gelöscht werden sollte. Der vorherige Befehl **lpq** verrät diese Nummer. Auf English werden die Druckaufträge Jobs genannt. Zum Beispiel löscht man mit **lprm 543** den Druckauftrag mit der Namen 543.

Wenn man nicht genau weiß wie ein bestimmter Drucker heißt gibt man **lpstat -a** ein und erfährt welche Drucker zur Verfügung stehen. Mit der Option **lpstat -d** gibt die Shell den Namen des Standarddruckers aus. Weitere Informationen über die Drucker gibt es mit den Optionen **-t** und **-s**.

3.12 Einfache reguläre Ausdrücke (Wildcards)

Die Suche nach Dateien und Verzeichnissen gestaltet sich manchmal recht schwierig, wenn man nicht genau weiß, wonach man sucht, oder wenn man gar mit dem Suchbegriff eine ganze Klasse Dateien zu erfassen sucht. Daraus folgt, dass es etwas wie Suchmuster geben muss. Diese Dinger heißen „Reguläre Ausdrücke“ oder „regular expressions“, abgekürzt regex. Eventuell sind sie bekannt vom DOS-Prompt:

- * bezeichnet eine beliebige Menge an Zeichen, mindestens aber ein Zeichen.

- ? bezeichnet ein beliebiges Zeichen - „?a*“ trifft für „Hallo“ zu aber nicht für „Spaten“.

- [...] bezeichnet ein Zeichen aus der Menge

Für die Beispiele legen wir zwei Dateien aa.txt und ab.txt an: `$ touch aa.txt ab.txt` ↗

- `$ echo *` ↗ Zeigt alle Dateien im Arbeitsverzeichnis an.

- `$ ls a*` ↗ Listet alle Dateien die mit a beginnen.

- `$ ls ?a.txt` ↗ Listet alle Dateien die mit beliebigen Zeichen beginnen gefolgt von a.txt.

- `$ rm *b*` ↗ Löscht alle Dateien die an b enthalten.

3.13 Netzwerk

Da es sich bei Linux um ein Netzwerkbetriebssystem handelt, könnte man über das Netzwerk ganze Bücher schreiben. Wir beschränken uns hier nur kurz auf die zwei wichtigsten Kommandos **ssh** und **scp**.

3.13.1 Secure Shell

Eine Shell kann man nicht nur lokal öffnen, man kann auch eine Shell auf einem anderen Rechner im Netzwerk öffnen und diese über das Netzwerk bedienen. Mit dieser Shell hat man dann die Möglichkeit, Kommandos auszuführen und Programme zu starten als würde man an diesem Rechner sitzen. Mit so einer „Remote Shell“ hat man also die selbe Befehlsmächtigkeit wie lokal.

Ein Programm, um eine „Remote Shell“ zu erlangen ist **ssh** (secure **shell**). Der Befehl **ssh infcip10** öffnet eine Remote Shell auf dem Rechner mit dem Namen infcip10. Bevor man diese „Remote Shell“ auch erhält, muss man sein Passwort eingeben.

Der Vorteil von ssh gegenüber anderen Programmen, die auch eine „Remote Shell“ starten, ist, dass ssh die Verbindung verschlüsselt, wobei die anderen sogar die Passwörter im Klartext übertragen.

Damit Programme mit graphischer Oberfläche auch gestartet werden können und diese Oberfläche lokal auf dem Bildschirm erscheint, muss eine Displayumleitung stattfinden. Das heißt, dass die Oberfläche aller Programme, die in dieser Shell gestartet werden, auf den lokalen Bildschirm „umgeleitet“ werden. Dies geschieht mit `ssh` automatisch oder, je nach Konfiguration, mit der Option **-X**: `ssh -XC infcip10` - öffnet eine Shell mit der Option X, die die Displayumleitung aktiviert. Die Option C komprimiert die Daten.

Falls man auf dem anderen Rechner einen anderen Login-Namen besitzt, kann man diesen angeben: `ssh goergen@infcip10` - öffnet eine Shell für den Benutzer goergen auf infcip10.

Wichtig: wenn Du von zu Hause auf die infcip-Rechner zugreifen willst, sollst Du sie mit ihren „vollen“ Namen ansprechen: `ssh user@infcip-remote.uni-trier.de`.

3.13.2 Secure Copy

`scp` ist im Grunde nichts anderes als ein Copy über eine `ssh`-Verbindung getunnelt, also ein verschlüsseltes Copy (secure **copy**).

```
scp QuellDatei User@Rechnername:ZielDatei
→ vom eigenen Rechner auf einen entfernten Rechner kopieren
scp User@Rechnername:QuellDatei Zieldatei
→ vom entfernten Rechner auf einen eigenen Rechner kopieren
scp User@Rechnername:QuellDatei User@Rechnername:ZielDatei
→ Dateien zwischen zwei entfernten Rechnern kopieren
man scp
→ Für mehr Informationen über SCP. :)
```

3.13.3 Zugriff von Windows aus (PuTTY, FTP-Client, Cygwin)

Da man natürlich auch von Windows aus gerne mal auf einen Linux/UNIX-Rechner zugreifen will, gibt es dafür auch nette Programme. Eins der besten dazu nennt sich PuTTY. Siehe: <http://www.chiark.greenend.org.uk/~sgtatham/putty/>

Achtung! PuTTY ist auch ein telnet-Client. Also darauf achten, dass Du als „Connection Type“ SSH wählst, denn die (unverschlüsselten und unsicheren!) Telnet-Verbindungen werden von unseren Rechnern nicht unterstützt.

Unter „Host Name“ `infcipXX.uni-trier.de` eingeben. **XX** soll man durch `-remote` oder eine Zahl zwischen 10 und 66 ersetzen. Port 22 wählen und auf Open klicken. Dann den Benutzernamen und das Passwort eingeben.

Mit einem FTP Client (z. B. FileZilla, WinSCP) lassen sich Dateien über SFTP leicht hoch- und runterladen. Als Host gelten alle CIP-Pool Rechner (`infcipXX.uni-trier.de`).

Für solche die sich unsterblich in die Shell verliebt haben, aber Windows weiter verwenden wollen, gibt es Cygwin <http://www.cygwin.com/>.

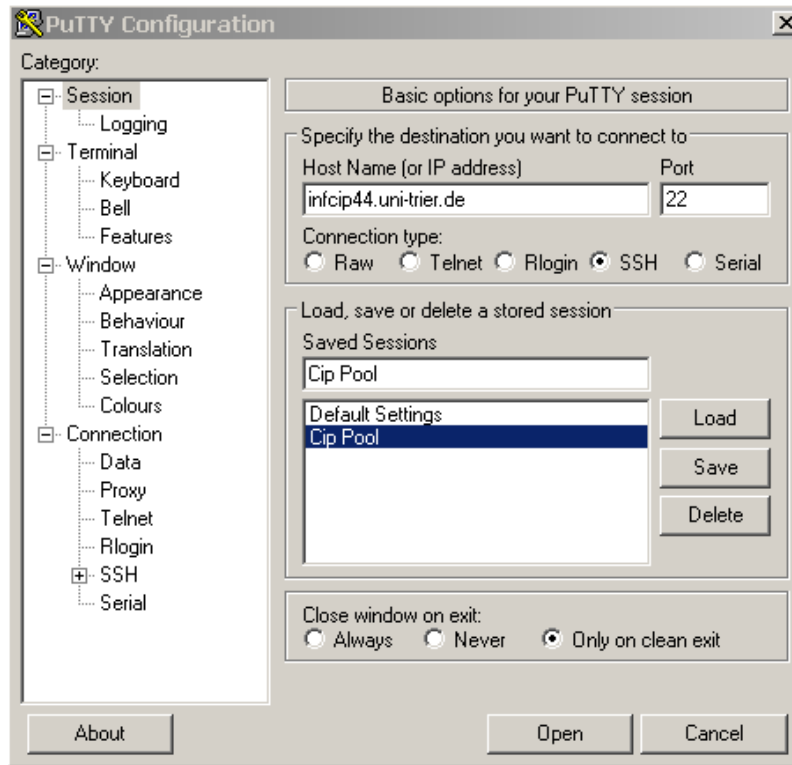


Abbildung 4: PuTTY Eingaben

3.14 Spielereien im CIP-Pool

- Suche dir einen Mitspieler und ladet euch Blobby Volley 2 aus dem Internet. Zu finden unter <http://sourceforge.net/projects/blobby>. Entpackt das Spiel per Shell in einem neuerstelltem Ordner. Jetzt nur noch in den Ordner wechseln und das Spiel mit `sh blobby` ausführen. Viel Spaß beim Spielen! Bitte beachtet jedoch: zu jeder Zeit ist Nutzern, die die Rechner zum Arbeiten verwenden wollen, den Vorzug zu geben.
- Suche im Internet nach „Brace Expansion“ und finde heraus wozu diese zu gebrauchen sind. Erstelle dann einen Ordner „Studium“, wechsele in den Ordner und erstelle mit einem Befehl sieben Unterverzeichnisse mit den Namen „1. Semester“, „2. Semester“, „3. Semester“, usw.
- Durchsuche alle Verzeichnisse nach Dateien mit dem Namen „klausur“. Nun suche nach allen Dateien deren Name „klausur“, „prüfung“ oder „pruefung“ enthalten.
- Starte die Shell und gib `vi` ein. Drücke die Taste `i` und gib folgenden Text ein „Ich will hier raus!“. Nun versuche `vi` zu verlassen. Das ist leider leichter gesagt als getan, im Internet findest du die Lösung aber wenn du gerade im CIP-Pool bist, kannst du dir auch das Plakat mit allen `vi`-Befehlen nahe der Tür anschauen und die Lösung finden.
- Logge dich auf einem Rechner ein und starte eine Shell. Nun benutze `ssh` um dich auf einem anderen Rechner einzuloggen auf dem du dich dann wieder auf einem anderen Rechner einloggst. Dann loggst du dich wiederum auf einem anderen Rechner ein. Schreibe ein Skript der dies automatisiert.

4 Sonstiges...

- ...zur Maileinrichtung (Thunderbird, Mutt, ...):
 - <http://cip.uni-trier.de/?id=2>
 - <https://cip.uni-trier.de/webmail> (Per Webmail Mails lesen/schreiben)
- ...zum Studium an der Uni:
 - <http://informatik.uni-trier.de> Fachbereich Informatik
 - <http://informatik.uni-trier.de/stundenplan> Stundenpläne der Informatik
 - <https://studip.uni-trier.de> Studip
 - <https://porta-system.uni-trier.de> Porta
 - <https://tricat.uni-trier.de> Suchportal der Bibliothek
 - <https://www.facebook.com/groups/informatik.unitrier> Info Trier Gruppe
- ...zu Linux auf diversen Webseiten:
 - <http://linuxcommand.org/tlcl.php> The Linux Command Line
 - <http://wiki.ubuntuusers.de/Shell> Allgemeines zur Shell
 - <http://www.washington.edu/computing/unix/vi> Vi Editor erklärt
 - http://openbook.galileocomputing.de/unix_guru Unix Guru
- ...zum Selbststudium durch Onlinevorlesungen:
 - <http://www.j3L7h.de/videos.html> Jörn Loviscach FH Bielefeld
 - <http://ttt.in.tum.de/lectures/index.php>
 - <http://timms.uni-tuebingen.de>
 - <http://www3.in.tum.de/teaching/bookDBMSeinf/aufzeichnungen/>
 - <http://webcast.berkeley.edu/> Empfehlenswert: Daniel Garcia CS10
 - <http://ocw.mit.edu> MIT OpenCourseWare
 - <https://www.rheinwerk-verlag.de/openbook/> Rheinwerk Openbook

Termine, Partys und Veranstaltungen

- 25. Oktober 2018 - Ersti Stadtrallye, Start um 18h an der Basilika
- 3. November 2018 ab 9h30 bis 18h00 - PAM 11 - Pi and More - piandmore.de
- 30. November 2018 ab 18h - UNiVerse LAN, Campus 2 (K101) lanparty.uni-trier.de
- Dezember 2018 - Vollversammlung und Weihnachtsfeier der Fachschaft Informatik

Autoren: Daniel Görgen, Zinaida Benenson, Alexander Greiml, Guillaume Kaufhold, Christophe Lange, Daniel Nofftz, Florian Reitz, Carsten Schmidt, Guido Schmitz, Nikolaj Schumacher, Markus Treinen, Marco Weber

Version: WiSe 2018