Kapitel 3. Betrieb

3.1. Unix-Grundlagen

Im folgenden finden Sie eine Einführung in die wichtigsten Unix-Kommandos sowie eine Beschreibung des Editors `vi`. Im Gegensatz zu anderen Betriebssystemen speichert Debian GNU/Linux alle nötigen Einstellungen in reinen Textdateien. Diese lassen sich mit den einfachsten Werkzeugen, wie zum Beispiel dem Editor vi, bearbeiten. Es werden keine Programme mit grafischer Benutzeroberfläche benötigt, der Systemadministrator hat 100%igen Zugriff auf alle Einstellungen, zur Not auch von außerhalb über eine simple Telefonleitung oder über das Internet via telnet oder besser ssh. Es werden die wichtigsten Grundlagen zur Systemadministration vermittelt, so daß Sie in jedem Fall Ihr System wieder zum Leben erwecken können.

Sicher gibt es einfacher zu bedienende Editoren als vi, dieser hat jedoch den Vorteil, schon direkt nach der Basisinstallation von Debian GNU/Linux vorhanden zu sein. Auch auf anderen Unix-Systemen trifft man auf diesen Standard-Editor, so daß man die Vorkenntnisse auch hier nutzen kann.

Da bisher keine grafische Oberfläche installiert wurde (diese ist beispielsweise für den Betrieb eines Servers nicht notwendig), finden Sie im folgenden ausschließlich textbasierte Programme.

Debian GNU/Linux stellt dem Systembetreiber aber auch zahlreiche grafische Werkzeuge zur Administration zur Verfügung, einige von ihnen werden später vorgestellt.

3.2. Allgemeines zum neuen System

Wenn Sie es bis hierher geschafft haben, Debian GNU/Linux auf Ihrem System zu installieren und das erste Mal zu starten, steht Ihnen nun ein lauffähiges Debian GNU/Linux-System zur Verfügung.

Alle installierten Programme wurden schon für Sie sinnvoll vorkonfiguriert. Dennoch ist es natürlich an vielen Stellen angebracht, dem System den letzten Schliff zu geben. Im folgenden erfahren Sie einiges über den Aufbau und die Funktionalität des Debian GNU/Linux-Systems, danach wird auf einige spezielle Programme eingegangen.

3.3. Ein Multiuser-, Multitasking-Betriebssystem

Debian GNU/Linux basiert auf dem Design der in den 60er Jahren entstandenen Unix-Systeme. Anders als die im täglichen Gebrauch – zu Hause oder in Büros – verbreiteten DOS-, Windows- und MacOS-Betriebssysteme, ist Unix im Serverbereich und auf Systemen mit vielen Benutzern, auf ein und demselben Rechner, verbreitet.

Dies bedeutet u.a., daß Debian GNU/Linux viele Funktionen von Hause aus mitbringt, die den anderen Betriebsystemen fehlen oder die zusätzlich erworben und installiert werden müssen. Debian erlaubt mehreren Benutzern die gleichzeitige Nutzung eines Rechners (Multiuser), hierzu ist es nötig mehrere, auch gleiche Programme, zur selben Zeit auszuführen. Diese Funktion nennt man Multitasking.

Ein Großteil der Komplexität und Leistungsfähigkeit von Unix-Systemen hat ihren Ursprung in diesen beiden Funktionen. Zum Beispiel muß das System bei mehreren Benutzern verhindern, daß ein Benutzer Dateien eines anderen versehentlich löschen kann.

Hat man dies einmal verstanden, fällt es viel leichter viele der Vorgänge und Eigenheiten eines Unix-Systems besser zu verstehen. Sie werden im folgenden lernen, diese beiden Funktionen sinnvoll zu nutzen.

3.4. Anmelden am System

Um Debian GNU/Linux zu benutzen, müssen Sie sich am System anmelden. Dies geschieht über die Eingabe eines sogenannten Usernamens und eines Paßwortes. Über diese Kombination kann das System feststellen, welche Zugriffsrechte der Benutzer hat und welche persönlichen Einstellungen zu verwenden sind.

Wenn Sie Debian GNU/Linux selber installiert haben, wurden Sie bereits bei der Installation nach einem Benutzernamen und einem Paßwort sowie einigen anderen Angaben gefragt. Möchten Sie sich an einem anderen System anmelden, fragen Sie den Systembetreuer nach einem Zugang.

Sie sollten in jedem Fall nicht auf die Idee kommen, für Ihren Zugang zu einem Linux-System auf ein Paßwort zu verzichten (auch wenn dies grundsätzlich möglich ist und Sie „sowieso nur alleine zu Hause“ an dem Computer arbeiten!). Ein Zugang ohne Paßwort steht für jeden anderen spätestens bei der ersten Verbindung ins Internet offen, bitte bedenken Sie das!

Auch ist es bei der Installation möglich, auf die Einrichtung eines normalen Benutzers zu verzichten. Dies mag in einigen wenigen Fällen sinnvoll sein, es ist aber meistens davon abzuraten. Der Superuser (root) hat generell alle Zugriffrechte an einem System. Debian GNU/Linux installiert aus Sicherheitsgründen einige Pakete so, daß die zugehörigen Programme nicht als Superuser ausgeführt werden können. Weiterhin ist es standardmäßig unter Debian GNU/Linux nicht ohne weiteres möglich, sich als Superuser über ein Netzwerk am System anzumelden. Es kann also zu Problemen und Verwirrung kommen, wenn Sie keinen normalen Benutzer angelegt haben.

Wenn Sie Ihr Debian GNU/Linux-System starten, sehen Sie eine Aufforderung zur Anmeldung am System. Je nach Konfiguration kann dies eine textbasierte oder eine grafische Anmeldung sein. Im einfachsten Fall sehen Sie in etwa folgendes:

```
Debian GNU/Linux potato book tty1

book login:
```

Bei einer unveränderten Debian-Installation steht in der ersten Zeile nach „Debian GNU/Linux“ der Name der Debian-Version (hier „potato“ für die Version 2.2), der Name des Rechners („book“) sowie Name und Nummer der Konsole, auf der Sie sich befinden.

Geben Sie nun Ihren Benutzernamen ein und drücken Sie RETURN, Sie werden dann nach Ihrem Paßwort gefragt.

```
Passwort:
```

Geben Sie hier Ihr gewähltes Paßwort ein. Beachten Sie, daß das Paßwort bei der Eingabe aus Sicherheitsgründen nicht angezeigt wird. Sollte die Anmeldung fehlschlagen, prüfen Sie, ob eventuell die Feststelltaste gedrückt wurde.

War die Anmeldung erfolgreich, sehen Sie eine kurze Willkommensnachricht, die in etwa so aussieht:

```
Linux book 2.0.36 #2 Sun Feb 21 15:55:27 EST 1999 i486 unknown

Copyright (C) 1993-1999 Software in the Public Interest, and others
```

```
Most of the programs included with the Debian GNU/Linux system are
freely redistributable; the exact distribution terms for each program
are described in the individual files in /usr/doc/*/copyright

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jul  3 21:53:40 on tty4.
You have mail.
$
```

Das letzte Zeichen, `$`, wurde von einem Programm, der sogenannten *shell*, ausgegeben. Deshalb nennt man dies auch den „shell-prompt“. Hier können Sie verschiedene Kommandos eingeben und so das System steuern oder Programme starten.

Als erster Versuch eignet sich das Kommando `whoami` gut.

Der Cursor rechts neben dem Shell-Prompt (meist ein blinkender Unterstrich (_)) zeigt an, an welcher Stelle Sie das nächste Zeichen auf der Tastatur eingeben sollen. Tippen Sie den Befehl `whoami` ein und drücken Sie RETURN.

`whoami` zeigt Ihren Benutzernamen an, Sie gelangen dann wieder zum Shell-Prompt zurück.

Wenn Sie Ihre Arbeit mit Debian beendet haben, sollten Sie sich wieder vom System abmelden. Hierfür können Sie den Befehl „exit“ benutzen. Eine weitere Möglichkeit ist das gleichzeitige Drücken der Tasten STRG+d, auch hiermit verlassen Sie die momentane *shell*, Ihre Benutzerumgebung.

Bedenken Sie, daß – wenn Sie sich nicht vom System abmelden – andere Leute Ihren Zugang mißbrauchen können, falls Sie sich von Ihrem Rechner entfernen!

3.5. Befehle wiederholen und ändern auf der Kommandozeile

Alles was Sie hinter einem Shell-Prompt eingeben, ist in irgendeiner Form Teil eines Kommandos. Die bei Debian GNU/Linux standardmäßig genutzte Shell (bash) hat verschiedene Funktionen, um das Eingeben und nachträgliche Ändern von Befehlszeilen zu erleichtern.

Sie können bereits eingegebene Kommandos wiederholen oder leicht verändern, um sie dann auszuführen. Probieren Sie es aus: Führen Sie irgendein Kommando, zum Beispiel `whoami` aus; drücken Sie dann die Taste PFEIL-OBEN. Der letzte Befehl erscheint wieder am Shell-Prompt. Sie brauchen nun nur noch RETURN zu drücken, um ihn noch einmal auszuführen.

Wenn Sie einige Befehle eingegeben haben, können Sie mit den Tasten PFEIL-OBEN und PFEIL-UNTEN diese Befehle noch einmal anzeigen lassen. Sie können so Befehle mehrfach ausführen oder Tippfehler nachträglich korrigieren, ohne die ganze Zeile nochmals eingeben zu müssen.

Am einfachsten können Sie den Cursor in der Befehlszeile mit den Pfeiltasten bewegen. Tippen Sie einfach mal einen Befehl, z.B. `whoami`, ein. Gehen Sie nun mit dem Cursor an die Stelle mit dem Tippfehler und löschen Sie mit BACKSPACE oder DELETE die gewünschten Zeichen. Geben Sie die noch fehlenden Zeichen ein und drücken Sie RETURN.

Es gibt noch viele leistungsstarke Funktionen in der shell. Eine detaillierte Beschreibung bekommen Sie mit dem Befehl `man bash`. Die komplette Dokumentation finden Sie unter `/usr/doc/bash/`.

An dieser Stelle möchten wir Ihnen nur einige, häufig benutze Funktionen vorstellen. Drücken Sie CTRL-a, der Cursor springt damit an den Anfang der Zeile. CTRL-k löscht von der aktuellen Position bis zum Ende die komplette Eingabe (k für „kill“). Probieren Sie dies mitten in einer längeren Zeile einmal aus. Die Kombination

CRTL-a, dann CTRL-k löscht die komplette Zeile. Die gelöschten Zeichen werden, unabhängig von der Länge, „gespeichert“ und können mit CTRL-y wieder an jeder Stelle eingefügt werden. Und zu guter Letzt: CTRL-e bringt den Cursor ans Zeilenende.

Spielen Sie einfach mal ein wenig mit diesen Funktionen herum: Sie werden schnell merken, daß sich mit ihnen sehr effizient arbeiten läßt!

3.6. Anmelden als Superuser (`root`)

Debian GNU/Linux ist als Multiuser-System dafür ausgelegt, mehrere Benutzer gleichzeitig bei der Arbeit zu unterstützen. Hierbei wurde besonderer Wert darauf gelegt, daß ein Programm eines Benutzers, wenn es denn mal abstürzt, nicht das komplette System abstürzen läßt. Über Zugriffrechte, die den Benutzern eines Systems von dem Systemadministrator zugewiesen werden, wird verhindert, daß wichtige Dateien im System von Unbefugten verändert oder gar gelöscht werden.

Manchmal ist es aber nötig, selbst wichtige Systemdateien zu verändern. Hierzu gehört die Installation von neuen Programmen (normalerweise über das Programm `dselect`) oder auch die Konfiguration des Netzwerkes. Ein solcher Eingriff ist ebenfalls nötig, wenn man eine serielle Maus angeschlossen hat und diese gegen ein Modell mit PS/2-Anschluß austauschen möchte.

Um diese Änderungen vornehmen zu können, müssen Sie mehr Zugriffsrechte zu Ihrem System erlangen als Ihnen als normaler Benutzer zur Verfügung stehen. Sie müssen „root“ werden, sich also als Superuser/ Systemadministrator am System anmelden.

Am einfachsten gelingt dies, wenn Sie sich bereits am Login-Prompt mit dem Benutzernamen root und dem dazugehörigen Paßwort anmelden. Das Paßwort haben Sie bereits bei der Installation gewählt, wenn Sie das System selber installiert haben. Arbeiten Sie an einem fremden System, bitten Sie den Systemadministrator, die gewünschten Einstellungen für Sie vorzunehmen.

Manchmal ist das Paßwort für den Superuser auch mehreren Personen bekannt, dies ist z.B. an Universitäten oder in Firmen üblich, um zu gewährleisten, daß möglichst immer ein Systembetreuer ansprechbar ist.

Melden Sie sich nun als Superuser (root) an. Überprüfen Sie mit dem Befehl `whoami` Ihre Identität. Loggen Sie sich möglichst bald wieder aus, wenn Sie als Superuser am System gearbeitet haben! Sie haben als Superuser alle Zugriffsrechte, um das gesamte System im schlimmsten Fall zu zerstören. Spielen Sie nicht mit dem System herum, solange Sie als Superuser angemeldet sind. Führen Sie nur die absolut notwendigen Arbeiten aus und melden Sie sich dann als normaler Benutzer wieder an!

Alternativ besteht auch die Möglichkeit, den Befehl `su` zu benutzen. Sie können so die Rechte eines anderen Benutzers erlangen, ohne sich am System ab- und wieder anmelden zu müssen.

Probieren Sie es einmal aus: Melden Sie sich mit Ihrem normalen Benutzernamen an (nicht root!). Geben Sie das Kommando `su` ein. Sie werden nun nach dem Paßwort für den Superuser gefragt: geben Sie es ein. Der Shell-Prompt sollte nun von `$` auf `#` wechseln. Sie können so leicht feststellen, daß Sie nun als Superuser angemeldet sind. Natürlich können Sie dies auch jederzeit wieder mit `whoami` überprüfen.

Sie können mit dem Befehl `su` auch die Identität jedes anderen Benutzers annehmen, solange Sie das Paßwort kennen oder als Superuser am System angemeldet sind. Benutzen Sie hierzu ebenfalls den Befehl `su` und geben dahinter (durch ein Leerzeichen getrennt) den Namen des Benutzers an. Beispiel: `su donald`.

Vielleicht werden Sie bemerken (nachdem Sie das Kommando `su` einige Zeit benutzt haben), daß nicht alle Einstellungen so sind, wie Sie vielleicht erwarten. Um auch mit dem Kommando `su` alle Einstellungen so vorzufinden, als ob man sich direkt von einem Login-Prompt angemeldet hätte, benutzen Sie den Befehl `su - donald`, also mit dem Zeichen `-` nach dem Kommando `su`.

3.7. Virtuelle Konsolen

Der Linux-Kernel unterstützt virtuelle Konsolen. Dies ist eine Methode, um Ihren Bildschirm und die Tastatur (sowie die Maus, falls das Programm `gpm` installiert wurde) so benutzen zu können, als wenn Sie an mehreren Geräten gleichzeitig arbeiten würden und diese alle mit dem gleichen Rechner-System verbunden wären.

Die Benutzung der virtuellen Konsolen ist sehr einfach. Über die Tastenkombinationen ALT+F1, ALT+F2, ALT+F3 bis ALT+F6 können Sie zwischen den verschiedenen virtuellen Bildschirmen umschalten. Probieren Sie es aus, indem Sie sich mehrfach einloggen und auf den verschiedenen Konsolen unterschiedliche Befehle ausführen.

Debian GNU/Linux ist standardmäßig für die Benutzung von sechs virtuellen Konsolen eingerichtet, auf diesen wird automatisch das Programm `login` gestartet. Sie können auf diese mit den Funktionstasten F1 bis F6 zugreifen (jeweils zusammen mit der ALT-Taste). Technisch sind auch noch mehr virtuelle Konsolen möglich, Debian GNU/Linux verwendet die siebte Konsole (ALT+F7) für das X-Window-System, die grafische Benutzeroberfläche.

Wenn Sie X benutzen, so verwendet dies automatisch die siebte virtuelle Konsole. Diese wird beim Start von X automatisch aktiviert, unabhängig davon, ob Sie das Kommando `startx` benutzen oder X über `xdm` oder ein anderes Programm starten.

Um von der grafischen Oberfläche X wieder auf die textbasierte Konsole zu wechseln, drücken Sie die Tastenkombination CTRL+ALT+F1 (oder +F2, +F3 usw. ...).

Sie müssen sich also nur merken, daß – wenn Sie von X auf eine Konsole wechseln möchten – zusätzlich die Taste CTRL zu drücken ist.

Wenn Sie sich einmal mit der Arbeitsweise der virtuellen Konsolen vertraut gemacht haben, werden Sie diese zu schätzen wissen. Sie können so schnell zwischen einem Editor und dem Compiler umschalten und auf der dritten Konsole noch die Logdateien im Auge behalten! Dies können Sie auch unter X erreichen, indem Sie mehrere Fenster öffnen, in vielen Fällen ist es aber gar nicht nötig, auf einem Rechner X zu installieren – beispielsweise bei einem Server.

Es ist sogar möglich, mehrere X-Server auf verschiedenen virtuellen Konsolen zu starten. Dies kann für den gleichzeitigen Betrieb von X mit verschiedenen Farbtiefen sinnvoll sein.

3.8. System herunterfahren

Schalten Sie Ihr Debian GNU/Linux-System niemals einfach aus! Sie riskieren in diesem Fall einen Verlust Ihrer Daten!

Wenn Sie Ihren Computer zu Hause benutzen, möchten Sie ihn vielleicht nachts abschalten (eigentlich schaltet kein richtiger Linux-Fan seinen Computer jemals ab, aber trotzdem wollen wir diesen Ausnahmefall kurz besprechen...).

Es ist eine sehr schlechte Idee, einen Linux-Computer nach der Arbeit einfach auszuschalten oder die RESET-Taste zu drücken. Der Linux-Kernel hat, um die Performance zu erhöhen, einen internen Festplatten-Cache. Das bedeutet, daß Informationen temporär im Speicher (RAM) des Computers abgelegt werden, bevor sie auf der Festplatte gespeichert werden. Die beschleunigt viele Aktionen stark. Periodisch werden diese Informationen auf die Festplatte gespeichert. Dies können Sie auch selber durch das Kommando `sync` erreichen.

Um Ihren Rechner ordnungsgemäß herunterzufahren, benutzen Sie bitte das Kommando `reboot` oder drücken Sie CTRL+ALT- und DEL-Tasten gleichzeitig. Debian GNU/Linux wird nun unmittelbar alle Programme beenden, alle Daten auf Festplatte speichern und den Rechner neu starten.

Um den Rechner abzuschalten, müssen Sie Superuser (root) sein. Benutzen Sie das Kommando `shutdown -h now`. Wenn Sie die Zeile `System halted, it's safe to turn off the computer.` sehen, können Sie den Rechner ausschalten.

Bei neueren Computern mit Advanced Power Management-Unterstützung (APM) und einem Kernel, der dies unterstützt, schaltet sich der Rechner selbsttätig ab.

Sie können aber auch als normaler User den Rechner jederzeit mit der Tastenkombination `CTRL-ALT-ENTF` herunterfahren, wenn der Superuser dies nicht deaktiviert hat (in `/etc/inittab`). Keine Angst, das System wird auch so korrekt runtergefahren.

3.9. Kommandozeile und Dokumentation

Einige einfache Beispiele für die Kommandozeile (alles das, was Sie hinter dem Shell-Prompt eingeben) wurden bereits ein paar Seiten zuvor besprochen. Nun beschäftigen wir uns mit etwas umfangreicheren Beispielen.

Eine minimale Kommandozeile enthält lediglich einen einzigen Befehl, ohne Parameter, z.B. `whoami`. Aber auch dies läßt sich noch ausbauen, geben Sie einfach mal `man whoami` ein.

Der Befehl `man` ruft die Bedienungsanleitung (manual page) für das Programm `whoami` auf. Mit der Taste SPACE können Sie in der Anleitung weiterblättern, die Taste `q` beendet das Programm `man`.

Hier ein noch erweitertes Beispiel:

```
man -k Postscript
```

Diese Kommando besteht aus 3 Teilen: Zuerst der eigentlich Name des Kommandos, `man`, gefolgt von einer sogenannten Option, - hier `-k`, abschließend dann das Argument `Postscript`.

Optionen verändern das Verhalten eines Programms. In den meisten Fällen werden Optionen mit dem Zeichen `-` eingeleitet. Die GNU-Programme kennen außerdem eine ausführliche Form der Optionen; für die Option `-k` wäre dies dann `--apropos`.

Probieren Sie dies nun einmal mit dem Befehl `man`, gefolgt von der Option `--h` und einmal in der ausführlichen Form mit der Option `--help` aus. Die Ergebnisse sind gleich.

Jedes Kommando hat seine eigenen Optionen. Es wird versucht, diese soweit möglich zu vereinheitlichen, so daß Sie die Optionen `-help` und `--version` bei allen GNU-Programmen antreffen sollten. Probieren Sie das ruhig einmal mit einigen verschiedenen Kommandos aus.

Aus historischen Gründen gibt es einige – manchmal geradezu bizarre – Abweichungen, die sich bis heute erhalten haben. So ist es z.B. möglich, bei den Kommandos `tar` oder `ps` das Zeichen „-“ vor den Optionen einfach wegzulassen.

Alle Zeichen, die nicht zu einer Option gehören und kein Kommandoname sind, nennt man Argumente. Argumente können verschiedene Zwecke erfüllen. Meistens handelt es sich um Namen von Dateien, die mit dem entsprechenden Kommando bearbeitet werden sollen. In dem oben genannten Beispiel (`man -k Postscript`) ist `Postscript` das Wort, welches vom Kommando `man` gesucht werden soll. Es wird dann nicht die Anleitung zum Programm `Postscript` gesucht, sondern es wird in allen Anleitungen nach dem Wort `Postscript` gesucht und alle Namen der entsprechenden Anleitungen angezeigt. Im Beispiel von `man whoami` ist `man` das Kommando und `whoami` das zu suchende Argument.

Wenn Sie nur wenige Programme auf Ihrem System installiert haben, sehen Sie wenige Suchergebnisse oder sogar nur die Meldung: `Postscript: nothing appropriate instead.`

3.9.1. Beschreibung der Kommandozeile

Die Kommandos, die Sie an einem Shell-Prompt eingeben können, folgen einer bestimmten Syntax. Wenn Sie beispielsweise `man man` eingeben, erhalten Sie die Anleitung (man page) zu dem Kommando `man`. In dieser Anleitung finden Sie weitere Beschreibungen zum Kommando `man` und den Optionen dieses Kommandos. Beispielsweise finden Sie dort:

```
man -k [-M path] keyword ...
```

Optionen, die in den eckigen Klammern ([]) stehen, sind optional und können ausgelassen werden. Sie können also den Befehl `man` auch ohne die Option `-M` benutzen, wenn Sie diese aber doch einsetzen, müssen Sie als Parameter den passenden Pfad angeben. Für die Option `-k` müssen Sie einen Suchbegriff (keyword) angeben. Die drei Punkte (...) bedeuten, daß Sie nach mehreren Begriffen suchen können. Trennen Sie diese durch ein Leerzeichen.

Hier noch ein Beispiel für eine etwas komplexere Beschreibung:

```
man  [-c|-w|-tZT  device]  [-adhu7V] [-m system[,...]] [-L
     locale] [-p string] [-M path] [-P pager] [-r  prompt]  [-S
     list] [-e extension] [[section] page ...] ...
```

Sie müssen nicht jede kleine Option verstehen – wichtig ist das Prinzip.

Neu ist in dieser Beschreibung das Zeichen `|`, es steht für „oder“ – nicht zu verwechseln mit der Verwendung dieses Zeichens in der Shell, hier ist nur die Man-Page gemeint. Sie können also eine der Optionen `-c`, `-w` oder `-tZT` benutzen, zusammen mit einem Device als Argument.

Gruppen von Optionen, wie `-adhu7V` bedeuten, daß Sie eine oder mehrere dieser Optionen gleichzeitig nutzen können. Dabei ist es nicht selbstverständlich, daß alle möglichen Kombinationen auch Sinn machen oder gar funktionieren. Lesen Sie dazu die komplette Anleitung zu `man`.

Beachten Sie weiterhin, wie die eckigen Klammern verschachtelt sind: `[[section] page]`. Dies bedeutet: wenn Sie eine `section` angeben, müssen Sie auch eine `page` angeben.

3.10. Dateien und Verzeichnisse

Dateien auf einer Festplatte dienen zur Organisation und Speicherung von Daten, ähnlich einem Blatt Papier. Dateien können in Verzeichnissen (oder „Schubladen“) geordnet sein. Im folgenden einige Informationen über die Organisation von Dateien und Verzeichnissen bei Debian GNU/Linux.

Das Zeichen `/` repräsentiert das sogenannte „root-“Verzeichnis. Alle weiteren Dateien und Verzeichnisse sind hier angeordnet. Wenn Sie vorher schon mal mit einem DOS- oder Windows-System gearbeitet haben, entspricht dies „in etwa“ dem Laufwerk C:. Machen Sie sich aber ab sofort mit dem Gedanken vertraut, daß es unter Linux keine Laufwerksbuchstaben gibt! Unter Linux finden Sie alle Laufwerke (Festplatten, CD-ROMs, Disketten...) unterhalb des „root-“Verzeichnisses (/).

Beispielsweise stellt `/home/fr` das User-Verzeichnis des Benutzers „fr“ dar. Unterhalb des „root-“Verzeichnisses findet sich auf jedem Debian GNU/Linux-System das Verzeichnis `home`. In diesem befinden sich weitere Unterverzeichnisse. Die Namensgebung dieser Verzeichnisse ist identisch mit den Benutzernamen, die beim Einrichten neuer Benutzer vergeben werden.

`/etc/X11/XF86Config`, dies ist die Konfigurationsdatei für das X-Window-System.

Wichtig: Linux unterscheidet Groß- und Kleinschreibung bei den Pfaden und Dateinamen.

Die Verzeichnisse sind in einer Struktur ähnlich einem Baum angeordnet. Vom „root-“Verzeichnis `/` verzweigt alles zu den weiteren Verzeichnissen. Unterhalb von `/` finden sich folgende Dateien und Verzeichnisse:

```
/
|-- System.map
|-- bin
|-- boot
|-- cdrom
|-- dev
|-- etc
|-- floppy
|-- home
|-- initrd
|-- lib
|-- lost+found
|-- mnt
|-- proc
|-- root
|-- sbin
|-- tmp
|-- usr
|-- var
|-- vmlinuz
```

Hier als Beispiel die weitere Verzweigung unterhalb von `/usr` (Sie finden hier hauptsächlich Programme, die von allen Benutzern ausgeführt werden können):

```
/usr/
|-- X11R6
|-- bin
|-- dict
|-- doc
|-- games
|-- i486-linuxlibc1
|-- include
|-- info
|-- lib
|-- local
|-- man
|-- openwin -> X11R6
|-- sbin
|-- share
‘-- src
```

Bis hierhin sollten Sie auf Ihrem System (je nachdem welche Pakete Sie installiert haben) in etwa die gleiche Struktur vorfinden. Deutliche Unterschiede von System zu System finden sich unterhalb von `/home`: die dortige Struktur ist abhängig von den Benutzern, für die sie angelegt wurde:

```
/home
|-- fr
|-- ftp
|-- geka
‘-- mw
```

Diese Verzeichnisse stellen die sogenannten „Home-Verzeichnisse“ der Benutzer dar, nach dem Anmelden am System befindet sich jeder Benutzer in seinem privaten Verzeichnis unterhalb von `/home`. In diesem Beispiel existieren die User `fr, geka` und `mw`. Das Verzeichnis `ftp` wurde im Laufe der Installation des FTP-Servers auf diesem Rechner eingerichtet. Dies stellt in diesem Sinne keinen eigentlichen Benutzer dar. Abweichend davon finden Sie das Home-Verzeichnis des Superusers (root) unterhalb von `/` als `/root/`. So können Sie für alle Benutzer des Systems die Homeverzeichnisse auf einer eigenen Partition halten. Für den Fall, daß das System beim Starten in einen unstabilen Zustand gerät und die Partition mit den Home-Verzeichnissen der Benutzer nicht mounten kann, besteht trotzdem für den Superuser die Möglichkeit, auf sein Home-Verzeichnis zuzugreifen.

Keines der Verzeichnisse unterhalb von `/` entspricht einem physikalischem Gerät, wie zum Beispiel einer Festplatte oder einem CD-ROM. Weiterhin verwendet Linux keine Buchstaben wie unter DOS zur Verwaltung dieser Geräte. Der Verzeichnisbaum stellt eine Abstraktion der vorhandenen Hardware dar, Sie können die Verzeichnisse benutzen, ohne Kenntnis von der eigentlichen Hardware zu haben. Alle Dateien Ihres Systems können auf einer einzigen Festplatte liegen oder auf vielen verschiedenen, einige davon in Ihrem Rechner, andere in anderen Rechnern irgendwo im Netzwerk.

Keine Panik, wenn Sie dies jetzt nicht völlig verstehen: die Idee dahinter ist anders als bei anderen Betriebssystemen. Es reicht wenn Sie sich merken, daß es keine Laufwerksbuchstaben gibt, Laufwerke – so wie Sie sie kennen – tauchen innerhalb des Verzeichnisbaumes auf, beispielsweise als `/cdrom` oder `/floppy`, wobei diese auch an jeder anderen Stelle liegen können, häufig unterhalb von `/mnt`.

3.11. Gruppen und Zugriffsrechte

Unix-Betriebssysteme, und damit auch Debian GNU/Linux, sind dafür ausgelegt, daß mehrere Benutzer zur gleichen Zeit am System arbeiten können. Dabei müssen bestimmte private Dateien vor anderen Benutzern geschützt werden, aber auch Systemdateien vor den Benutzern geschützt werden. Sie können sie sehr leicht selber überprüfen:

Melden Sie sich mit Ihrem Benutzernamen am System an, benutzen Sie nicht den Zugang des Superusers (root), und geben Sie das Kommando: `rm /etc/resolv.conf` ein.

```
bash-2.03$ rm /etc/resolv.conf
rm: remove write-protected file ‘/etc/resolv.conf’? y
rm: cannot unlink ‘/etc/resolv.conf’: Permission denied
```

Das System schützt diese Datei vor Veränderungen durch andere Benutzer als dem Superuser. Wenn jeder Benutzer Veränderungen an wichtigen Systemdateien vornehmen könnte, würde dies schnell zu Problemen führen. Sehen wir uns die Datei einmal etwas näher an:

Geben Sie nun das Kommando `ls -l /etc/resolv.conf` ein. Sie bekommen diese Ausgabe:

```
-rw-r--r-- 1 root root 119 Nov 02 1999 /etc/resolv.conf
```

Die Option `-l` des Kommandos `ls` gibt den Dateinamen sowie alle weiteren Informationen zu der Datei aus. Diese Informationen sind ziemlich einfach zu verstehen. Die Größe der Datei ist 119 Byte, die Datei wurde zuletzt am 02. November 1999 geändert und der Dateiname ist `/etc/resolv.conf`. Weiter links wird die Sache etwas komplizierter...

Kurz und knapp: `-rw-r--r--` steht für die eigentlichen Zugriffsrechte der Datei, die `1` steht für die Anzahl der (hard) Links auf diese Datei (oder die Anzahl der Dateien in einem Verzeichnis) und `root root` bezeichnet den Besitzer sowie die Gruppe, zu der die Datei gehört.

Doch nun etwas ausführlicher...

3.11.1. Gruppen

Jede Datei auf Ihrem Debian GNU/Linux-System hat zwei Eigentümer: einen User und eine Gruppe. Das oben angeführte Beispiel ist da etwas verwirrend, es gibt sowohl einen User als auch eine Gruppe `root`. Gruppen sind, wie auch im echten Leben, Ansammlungen von Personen, sprich Benutzern auf einem System. Diese Mitglieder einer Gruppe können gemeinsamen Zugriff auf bestimmte Dateien haben, beispielsweise auf alle Dateien unterhalb von `/var/www/projekte/debian/`, wenn sie gemeinsam an den Webseiten zu einem Debian-Projekt arbeiten sollen. Sie können auch beispielsweise bestimmte Benutzer der Gruppe `dialout` (bedeutet soviel wie „rauswählen“) zuordnen, damit diese per Modem eine Verbindung ins Netz herstellen können.

Das Kommando `groups` zeigt Ihnen an, zu welchen Gruppen Sie gehören. Dies ist abhängig von dem Benutzernamen, mit dem Sie sich am System angemeldet haben.

Sehen Sie sich nun die Datei `/etc/group` an, benutzen Sie hierzu beispielsweise das Kommando `more` (`more /etc/group`). Beachten Sie die Gruppe `root`, in dieser sollte als einziger der Benutzer `root` eingetragen sein sowie Ihre eigene Gruppe, auch hier sollten nur Sie eingetragen sein. Es gibt einige weitere Gruppen in dieser Datei, beispielsweise `dialout` (siehe oben), `floppy` – diese Benutzer können auf das Diskettenlaufwerk zugreifen und andere. Nach der Installation sind keine weiteren Benutzer in den verschiedenen Gruppen aufgeführt, dies ist die Aufgabe des Systemverwalters, also Ihre `;-)`. Hier ein Beispiel für eine veränderte Datei `/etc/group` aus einem laufenden System:

```
root:x:0:
daemon:x:1:fr
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:lp
mail:x:8:fr,geka,mw
news:x:9:
uucp:x:10:
proxy:x:13:
kmem:x:15:
dialout:x:20:fr
fax:x:21:fr
voice:x:22:fr
cdrom:x:24:fr
floppy:x:25:fr
```

```
tape:x:26:fr
sudo:x:27:fr
audio:x:29:fr
dip:x:30:
majordom:x:31:majordom
postgres:x:32:
www-data:x:33:fr
backup:x:34:fr
msql:x:36:fr
operator:x:37:fr
list:x:38:fr
irc:x:39:fr
src:x:40:fr
gnats:x:41:
shadow:x:42:
utmp:x:43:telnetd
video:*:44:
staff:x:50:fr
games:x:60:fr
qmail:x:70:
users:x:100:fr
telnetd:x:101:
fr:x:1000:
mw:x:1001:
geka:x:1002:
fr2:x:1003:
nogroup:x:65534:
mysql:x:102:
```

Weitere Informationen zu dieser Datei bekommen Sie mit dem Kommando `man group`.

Mit dem Kommando `ls -l /home` können Sie sich einen Überblick über die Stammverzeichnisse aller Benutzer auf dem System verschaffen. Jedes Verzeichnis sollte auch dem dazugehörigen Benutzer gehören. Wenn Sie das System neu installiert haben, werden Sie der einzige Benutzer sein. Deshalb auch hier ein Beispiel aus der Praxis:

```
bash-2.03$ ls -l /home/
total 8
drwxr-sr-x   65 fr       fr           5120 Jan 19 02:48 fr
dr-xr-xrwx   45 fr       fr           1024 Jan  9 22:44 ftp
drwxr-sr-x   12 geka     geka         1024 Jan 19 03:20 geka
drwxr-sr-x    5 mw       mw           1024 Jan  4 08:27 mw
```

3.11.2. Zugriffsrechte

Neben dem Besitzer und der Gruppe, zu denen eine Datei gehört, verfügt jede Datei auch über Zugriffsrechte, über die festgelegt wird, wer diese Datei lesen, schreiben oder ausführen darf. Es gibt noch weitere Details, die wir aber übergehen wollen.

Wie wir schon vorher gesehen haben, werden die Zugriffsrechte bei dem Kommando `ls -l` ganz links an den ersten zehn Stellen angezeigt. Die erste Stelle hat nicht direkt etwas mit den Zugriffsrechten zu tun, sie zeigt

vielmehr den Dateityp an. Ein „-“ steht für eine normale Datei, ein `d` kennzeichnet ein Verzeichnis und ein `l` steht für einen Link.

Die weiteren neun Stellen lassen sich in drei Gruppen teilen. Dies sind von links nach rechts: der Besitzer der Datei (owner), die Gruppe (group) und schließlich die Allgemeinheit (world). Jeder dieser drei Gruppen gehören drei dieser neuen Stellen. Jeder dieser drei Stellen steht für `r` lesen (read), `w` schreiben (write) und `x` ausführen (execute).

Im Detail bedeuten die drei Buchstaben r, w und x folgendes:

r – lesen : bei Dateien kann der Inhalt der Datei gelesen werden. Bei Verzeichnissen kann man den Inhalt des Verzeichnisses auflisten lassen.

w – schreiben : bei Dateien kann diese Datei verändert und gespeichert werden. Bei Verzeichnissen können neue Dateien angelegt und bereits bestehende Dateien gelöscht werden.

x – ausführen : Dateien können als Kommando ausgeführt werden. Dies macht nur Sinn, wenn diese Datei wirklich ein Kommando darstellt. Sie können eine Grafik ausführbar machen, es ergibt aber keinen Sinn. Da Verzeichnisse nicht ausgeführt werden können, bedeutet hier ein gesetztes `x`, daß Sie in dieses Verzeichnis wechseln können. Um also in einem Verzeichnis mit Dateien arbeiten zu können, benötigen Sie die Kombination `x` und `r` sowie gegebenenfalls auch `w`.

Für Verzeichnisse ist dies alles ein wenig verwirrend, daher hier einige Beispiele:

`r--`Eigentümer, Gruppe oder andere können den Inhalt dieses Verzeichnisses auflisten. Die Dateien selber in dem Verzeichnis können gelesen, gelöscht oder verändert werden.

`r-x` Dieser Modus erlaubt das Auflisten der Dateien in dem Verzeichnis und gibt den Zugriff auf die Dateien frei. Sie können allerdings keine neuen Dateien anlegen oder bestehende Dateien löschen. Das Ansehen und Verändern von Dateien ist erlaubt, Programme können ausgeführt werden, wenn dies von den Rechten der Dateien selber her erlaubt ist.

`--x` Sie können auf die Dateien in dem Verzeichnis zugreifen, diese aber nicht auflisten. Sie müssen also wissen welche Dateien sich in dem Verzeichnis befinden, um auf diese zugreifen zu können.

`rwx` Sie können alles mit den Dateien anstellen, solange die Rechte der Dateien selber dies zulassen.

Daraus folgen einige interessante Tatsachen, die Sie beachten sollten:

Schreibrechte auf einem Verzeichnis entscheiden darüber, ob Sie eine Datei in einem Verzeichnis löschen dürfen. Eine Datei deren Rechte auf nur-lesen gesetzt sind, kann gelöscht werden, wenn Sie die nötigen Rechte haben um in diesem Verzeichnis zu schreiben! Weiterhin können Sie eine Datei in einem Nur-lesen-Verzeichnis nicht löschen, auch wenn Sie die nötigen Zugriffsrechte auf die Datei selber haben.

Dies bedeutet auch, daß Sie, wenn Sie der Besitzer eines Verzeichnisses sind, auch die Dateien darin löschen können, auch wenn diese dem Superuser (root) gehören.

Zugriffsrechte auf ein Verzeichnis haben also auch direkten Einfluß auf die Dateien in diesem Verzeichnis, an dieser Stelle kommen die Zugriffsrechte auf Dateien ins Spiel. Wenn Sie keinen Zugriff auf das Verzeichnis haben, spielen auch die Rechte an den Dateien für Sie keine Rollen, Sie kommen ja eh nicht an die Dateien...

3.11.2.1. Einige Beispiele

Um die Zugriffsrechte von Dateien und Verzeichnissen zu verändern, steht das Kommando `chmod` unter Debian GNU/Linux zur Verfügung. Spielen wir einmal ein wenig damit herum:

Erzeugen Sie zunächst eine neue Datei, beispielsweise mit dem Kommando `touch testdatei`. Das Kommando `touch` wird normalerweise dazu benutzt, die Datei mit einem aktuellen „Zeitstempel“ zu versehen.

Wenn Sie jedoch einen Dateinamen angeben, der noch nicht existiert, so wird diese Datei neu angelegt, mit einer Länge von 0 Byte. Überprüfen Sie dies mit dem Kommando `ls -l` und werfen Sie einen Blick auf die Zugriffsrechte:

```
bash-2.03$ touch testdatei
bash-2.03$ ls -l testdatei
-rw-r--r--    1 fr       fr              0 Jan 19 18:15 testdatei
```

Bei Ihrem Versuch wird die Datei natürlich einen anderen Zeitstempel haben und Benutzer- und Gruppenzugehörigkeit entsprechen Ihrem Loginnamen. Die Zugriffsrechte (`-rw-r--r--`) werden von Debian GNU/Linux automatisch für neue Dateien auf die gezeigten Werte gesetzt. Sie können diese Vorgabe mit dem Kommando `umask` ändern.

Sehen Sie sich zunächst die Man-Page zu `chmod` mit dem Kommando `man chmod` an. Wir werden hier nicht auf jedes Detail eingehen, sondern an einigen Beispielen zeigen, wie sich `chmod` mit verschiedenen Parametern auswirkt.

Führen Sie das Kommando `chmod u+x testdatei` aus. Sehen Sie sich die Veränderung mit `ls -l testdatei` an. Es wurden Rechte zum Ausführen (`x` - execute) der Datei für den Besitzer (`u` - User) hinzugefügt (`+` - Pluszeichen).

```
bash-2.03$ chmod u+x testdatei
bash-2.03$ ls -l testdatei
-rwxr--r--    1 fr       fr              0 Jan 19 18:15 testdatei
```

Ein solches Kommando können Sie beispielsweise auf ein selbstgeschriebenes Shellscript oder Perl-Programm anwenden, damit es auch ausführbar ist.

Wenn Sie nun noch möchten, daß niemand außer Ihnen einen Blick in Ihre Arbeit werfen kann, so müssen Sie die Rechte zum Lesen der Datei für die Gruppe (`g` - Group) sowie alle anderen Benutzer (`o` - Other) entfernen (`-` - Minuszeichen). Sie können dies mit dem Kommando `chmod go-r testdatei` erreichen:

```
bash-2.03$ chmod go-r testdatei
bash-2.03$ ls -l testdatei
-rwx------    1 fr       fr              0 Jan 19 18:15 testdatei
```

Wie Sie gesehen haben, können Sie mit den Zeichen `+` (Minus) oder `-` (Minus) Rechte hinzufügen oder entfernen. Manchmal ist es damit etwas verwirrend, einen gewünschten Zustand herzustellen. Daher bietet `chmod` noch die Option `=` (Gleichheitszeichen), welche genau die angegebenen Rechte setzt und alle anderen löscht. Auch hier können Sie wieder die Buchstaben `ugo` (User, Group, Other) benutzen:

```
bash-2.03$ chmod ugo=rx testdatei
bash-2.03$ ls -l testdatei
-r-xr-xr-x    1 fr       fr              0 Jan 19 18:15 testdatei
```

Die Datei ist nun für jeden Benutzer lesbar und kann auch von jedem ausgeführt werden. Weiterhin kann keiner der Benutzer diese Datei schreiben.

Entfernen Sie nun die Rechte zum Ausführen der Datei für alle Benutzer (`chmod a-x testdatei`), bei einer Testdatei brauchen wir diese nicht.

Versuchen Sie einmal die Datei zu löschen. Zur Erinnerung: Sie hatten vor kurzem die Datei mit dem Kommando `chmod ugo=rx testdatei` behandelt. Löschen Sie also die Datei mit dem Kommando `rm testdatei`:

```
bash-2.03$ rm testdatei
rm: schreibgeschützte Datei »testdatei« entfernen?
```

Wenn Sie die Umgebungsvariablen nicht passend gesetzt haben, wird Ihnen die Fehlermeldung in englischer Sprache präsentiert.

Da Sie (und auch kein anderer) keine Rechte haben, die Datei zu schreiben, fragt das Kommando `rm`, ob Sie diese Aktion wirklich durchführen wollen. Dies ist eine spezielle Funktion von `rm` und hat eigentlich wenig mit den Zugriffsrechten zu tun. Wenn Sie die Datei wirklich löschen möchten, können Sie die Frage bestätigen. Stören Sie sich nicht an der Ausgabe der Rechte an Zahlenform, des Kommandos `rm`, Sie können die Bedeutung in der Man-Page nachlesen.

3.12. Orientierung innerhalb von Debian

Es gibt einige Unterschiede zwischen Debian GNU/Linux und anderen Distributionen. Selbst wenn Sie Linux und andere Distributionen bereits kennen, gibt es einige Dinge, die Sie wissen sollten, um Ihr System in einem gut zu wartendem Zustand zu halten. Dieser Abschnitt dient Ihrer Orientierung.

Das wichtigste Konzept, das man verstehen muß, ist die Paketverwaltung von Debian. Im wesentlichen müssen Sie akzeptieren, daß große Teile Ihres System unter der Kontrolle der Paketverwaltung stehen. Sie können nicht so ohne weiteres „von Hand“ (zum Beispiel wenn Sie eine eigene Version des Apache-Servers aus den Sourcen übersetzt haben) ein Paket aktualisieren. Diese Aufgabe nimmt Ihnen das Debian GNU/Linux-Paketsystem ab. Nutzen Sie dies, um Pakete zu aktualisieren. Folgende Bereiche stehen unter der Kontrolle dieses Paketsystems:

- /usr (mit Ausnahme von /usr/local)
- /var (Sie können sich ggf. /var/local anlegen)
- /bin
- /sbin
- /lib

Wenn Sie zum Beispiel `/usr/bin/perl` ersetzen, weil Sie auf einer CD oder auf einem FTP-Server eine aktuellere Version wie auf Ihren Debian GNU/Linux- CDs gefunden haben (kaum zu glauben, daß sowas wirklich passiert...), wird das zunächst funktionieren.

Wenn Sie nun jedoch ein Paket mit `dselect` installieren, welches ebenfalls irgendeine Perl-Version benötigt, so wird dselect immer die Version heranziehen, die auf Ihrem Installationspfad enthalten ist. Das heißt, daß dselect unter Umständen eine ältere Version installiert, obwohl Sie schon eine neuere Version übersetzt hatten. Bitte beachten Sie das!

Aktualisieren Sie jedoch Ihr Perl-Paket, über `dpkg`/`dselect` oder `apt`, dann wird die Datei durch die aus dem Debian-Paket ersetzt. Erfahrene Anwender können dieses verhindern, indem Sie das entsprechende Paket auf hold (in der Paketauswahl von `dselect` mit der Taste `=`) setzen oder `dpkg-divert` benutzen.

Wenn Sie bestimmte Pakete durch eigene, modifizierte Versionen ersetzen wollen, sollten Sie sich intensiv mit dem Debian GNU/Linux-Paketmanagement befassen.

Eine der wichtigsten Vereinbarungen, an den Sie sich gewöhnen müssen, ist, daß sich alle Konfigurationsdateien unterhalb von `/etc/` befinden. Hierbei ist es vielfach so, daß für einzelne Programme

zusätzliche Verzeichnisse bei der Installation erzeugt werden, meist geschieht dies für Programme, die über mehr als eine Konfigurationsdatei verfügen, beispielsweise für den Webserver `apache`.

Unter `/usr/doc/` beziehungsweise `/usr/share/doc/` finden Sie die Dokumentation zu den auf Ihrem System installierten Paketen. Zu vielen Paketen finden Sie in dem entsprechenden Verzeichnis eine Datei `README.Debian` (oder ähnlich), welche die speziellen Anpassungen an Debian GNU/Linux beschreibt.

Weiterhin befinden sich in jedem Verzeichnis auch die jeweiligen Lizenzen zu den Paketen.

3.13. Arbeiten mit Dateien – Mini-Workshop

Um mit Ihrem System arbeiten zu können, müssen Sie etwas über das Erzeugen, Verschieben, Umbenennen und Löschen von Dateien und Verzeichnissen erfahren.

3.13.1. `pwd` – print working directory

Zunächst ist es jedoch wichtig zu wissen, an welcher Stelle des Verzeichnisbaumes man sich befindet. Wie schon beschrieben, befinden Sie sich nach dem Anmelden am System in Ihrem Home-Verzeichnis. Sie können dies mit dem Kommando `pwd` überprüfen. Die Ausgabe sollte in etwa so aussehen:

```
bash-2.02$ pwd
/home/fr
```

Wobei auf Ihrem System statt `fr` Ihr eigener Benutzername, mit dem Sie sich angemeldet haben, erscheint.

3.13.2. `ls` – list

Dieses Kommando zeigt, wenn es ohne weitere Parameter verwendet wird, alle Dateien in dem aktuellen Verzeichnis an. Nach dem Anmelden an einem neu installierten Debian GNU/Linux-System befinden Sie sich in Ihrem Home-Verzeichnis, und dieses ist leer. Das Kommando `ls` wird also nichts anzeigen. (Das Verzeichnis ist nicht wirklich leer, Sie sehen lediglich mit dem Kommando `ls` ohne Parameter nicht die angelegten Dateien.)

Sie können aber auch das Kommando `ls` mit einem Pfad als Option aufrufen. Beispielsweise zeigt `ls /` das „root-“Verzeichnis des Systems an.

3.13.3. `cd` – change directory

Mit diesem Kommando können Sie in ein anderes Verzeichnis (directory) wechseln. `cd /tmp` wechselt beispielsweise in das Verzeichnis für temporäre Dateien unterhalb von `/`.

Auf zwei Besonderheiten möchte ich an dieser Stelle eingehen, die nicht nur mit dem Kommando `cd` funktionieren, aber zum besseren Verständnis hier gut untergebracht sind.

Das Zeichen ~ steht als Abkürzung für den kompletten Pfad zu Ihrem privaten Home-Verzeichnis. `cd` mit der Option ~, also `cd ~`, wechselt ins Home-Verzeichnis.

Weiterhin möchte man häufig in ein Verzeichnis in der Struktur höher wechseln. Sicher könnte man mit `pwd` nachsehen, wo man sich gerade befindet und dann dem Kommando `cd` den passenden Pfad übergeben. Als

einfache Alternative steht aber das Kürzel `..` für das übergeordnete Verzeichnis zur Verfügung. `cd ..` wechselt also ins übergeordnete Verzeichnis (beachten Sie das Leerzeichen).

Eine weitere Abkürzung stellt `.` dar. Diese steht für das aktuelle Verzeichnis, in dem Sie sich gerade befinden, doch dazu gleich in einem anderen Beispiel.

3.13.4. `mkdir` – make directory

Mit diesem Kommando können Sie weitere Verzeichnisse anlegen. Wechseln Sie in Ihr Home-Verzeichnis mit `cd ~` und erzeugen Sie ein Verzeichnis `test` (mit `mkdir test`). Überprüfen Sie mit `ls`, ob es funktioniert hat. Erzeugen Sie nach Belieben einige weitere Verzeichnisse, auch unterhalb von `test`.

3.13.5. `cp` – copy

Mit diesem Kommando können Sie Kopien von Dateien erzeugen. Kopieren Sie die Datei `/etc/profile` in Ihr Home-Verzeichnis mit `cp /etc/profile .` (Hier also das versprochene Beispiel mit nur einem Punkt.) Prüfen Sie mit `ls`, ob sich in Ihrem Verzeichnis nun eine Datei `profile` befindet.

3.13.6. `more` – Anzeigen von Dateien

`more profile` zeigt Ihnen den Inhalt der Datei `profile` seitenweise auf der Konsole an. Sie können mit der SPACE-Taste seitenweise weiterblättern und mit der Taste `q` das Programm `more` wieder verlassen. `more` zeigt am unteren Bildschirmrand die aktuelle Position in der Datei in Prozent an. Am Ende einer Datei wird `more` automatisch beendet. `more` kann, wie die meisten anderen Programme auch, über Optionen auf der Kommandozeile gesteuert werden. Sie können diese Optionen auch in die Umgebungsvariable `MORE` schreiben, die Optionen werden dann bei jedem Aufruf von `more` benutzt.

Lesen Sie die man-page zu `more`, das Programm hat noch einige andere interessante Möglichkeiten.

Tip: Sollten Sie auf eine gepackte Datei stoßen, so können Sie diese mit dem Kommando `zmore` ansehen.

Wenn Ihnen die Optionen von `more` nicht ausreichen, sollten Sie einen Blick auf das Programm `less` werfen. Es verfügt über die gleichen Funktionen, wurde aber noch um einige nützliche Funktionen erweitert.

3.13.7. `mv` – move

Zum Verschieben von Dateien benötigt das Kommando `mv` zwei Parameter: der erste Parameter ist die Quelldatei, der zweite die Zieldatei. Verschieben Sie die Datei `profile` in das erstellte Verzeichnis `test`, mit `mv profile test`. `mv` dient aber auch zum Umbenennen von Dateien. Wechseln Sie in das Verzeichnis `test` und benennen Sie die Datei `profile` in `testdatei` um (`mv profile testdatei`).

Weiterhin können Sie mit `mv` auch Verzeichnisse verschieben oder umbenennen, die Syntax unterscheidet sich dabei nicht, egal ob Sie mit Dateien oder Verzeichnissen arbeiten.

3.13.8. `rm` – remove

Mit dem Kommando `rm` können Sie eine oder mehrere Dateien löschen. Im einfachsten Fall geben Sie den Namen der zu löschenden Datei an, also: `rm ichwillweg.txt` (Sollte diese nicht existieren, so können Sie diese mit dem Kommando `touch ichwillweg.txt` anlegen). Als Parameter kann dem Kommando `rm` eine Reihe von Dateinamen mitgegeben werden, diese werden durch Leerzeichen getrennt: `rm ichwillweg.txt ichauch.txt metoo.asc removeme.txt done.sh`, dieses Kommando löscht fünf Dateien von der Platte. An dieser Stelle ein wichtiger Hinweis: Es gibt kein `undelete` unter Linux. Dateien, die Sie mit `rm` gelöscht haben, können Sie nicht zurückholen. (Es gibt Programme, die auch mit dem zur Zeit aktuellen `ext2`-Dateisystem gelöschte Dateien zurückholen können. Dies ist aber momentan noch in der Entwicklung. Besser ist es, auf ein „journaling Filesystem“ für Linux zu warten, ext3, XFS oder reiserFS sind da zu nennen, diese werden solche Funktionen besser unterstützen.)

Wenn Sie ein Verzeichnis inklusive aller darin enthaltenen Dateien löschen möchten, können Sie dies auch mit dem Kommando `rm` erledigen. Hierzu dient die Option `-rf`. Ein Beispiel: `rm -rf /home/fr/test/` löscht im Home-Verzeichnis das Verzeichnis `test` mit allen Dateien und Unterverzeichnissen. Etwas, was Sie nicht ausprobieren sollten (man findet das manchmal, weil sich Leute einen Spaß daraus machen...). Als weiteres Beispiel folgendes (nicht abtippen!!!): `rm -rf /` (nicht abtippen!!!). Wie schon beschrieben stellt `/` das Startverzeichnis des gesamten Verzeichnisbaums dar. Sie würden also Ihr komplettes System von der Platte verbannen.... Also nicht auf diesen kleinen Spaß hereinfallen...

3.13.9. `rmdir` – remove directory

Nun fehlt uns noch ein Kommando, um lediglich ein Verzeichnis, in dem sich keine Dateien befinden, zu entfernen. `rmdir` mit dem Verzeichnisnamen erledigt dies für uns. Natürlich läßt sich auch `rm` dafür nutzen, mit den entsprechenden Optionen, Sie können selbst entscheiden, welches Kommando Sie benutzen wollen... Noch schnell ein Beispiel: `rmdir test` entfernt das Verzeichnis `test` im aktuellen Verzeichnis.

3.13.10. Versteckte Dateien (`.datei`)

Namen von versteckten Dateien oder Verzeichnissen beginnen mit einem Punkt (.). Sie können das Kommando `ls` mit der Option `-a` dazu bringen, auch diese versteckten Dateien anzuzeigen. Sie können das sehr einfach in Ihrem Homeverzeichnis ausprobieren, dort werden bei der Einrichtung eines neuen Benutzers und später durch verschiedene Programme diverse versteckte Dateien und Verzeichnisse angelegt.

Wechseln Sie in Ihr Homeverzeichnis (mit dem Kommando `cd`) und sehen Sie sich den Inhalt des Verzeichnisses einmal an, inklusive der versteckten Dateien (mit dem Kommando `ls -la`). Sie sehen nun die „normalen“ Dateien sowie auch die versteckten Dateien. Dabei wird Ihnen vielleicht auffallen, daß es zwei etwas außergewöhnliche Dateien, nämlich `.` und `..` gibt. Diese stellen das aktuelle Verzeichnis „.“, in dem Sie sich befinden, sowie das übergeordnete Verzeichnis `..` dar. Wenn Sie das Kommando `ls` mit der Option `-lA` benutzen, werden diese beiden Dateien nicht mit angezeigt.

Der Grund für versteckte Dateien liegt nicht in der Geheimhaltung von Daten. Vielmehr ist es im täglichen Umgang mit dem System nicht sinnvoll, alle möglichen Dateien anzuzeigen, die sich in Ihrem Homeverzeichnis befinden. Viele Programme legen dort auch individuelle Konfigurationsdateien ab, es hat sich eingebürgert, diese Dateien oder Verzeichnisse mit einem Punkt beginnen zu lassen, so daß diese nicht bei der normalen Arbeit mit Dateien stören.

Über diese Konfigurationsdateien in Ihrem Verzeichnis können Sie das Verhalten oder Aussehen von Programmen verändern. Diese Änderungen sind nur wirksam, wenn Sie sich mit Ihrem Benutzernamen am

System angemeldet haben. Systemweite Konfigurationsdateien finden Sie im Verzeichnis `/etc/`.

3.13.11. `find & locate` – Finden von Dateien

Um Dateien in Ihrem System zu finden, stehen Ihnen auf der Kommandozeile zwei Programme zur Verfügung: `find` und `locate`. Mit dem Programm `find` können Sie die Festplatte nach Dateien durchsuchen, dies kann je nach Größe der Platten einige Zeit dauern. `find` verfügt über einige Parameter, die Sie in der Man-Page nachlesen können.

```
bash-2.03$ find --help
Usage: find [path...] [expression]
default path is the current directory; default expression is -print
expression may consist of:
operators (decreasing precedence; -and is implicit where no others are given):
      ( EXPR ) ! EXPR -not EXPR EXPR1 -a EXPR2 EXPR1 -and EXPR2
      EXPR1 -o EXPR2 EXPR1 -or EXPR2 EXPR1 , EXPR2
options (always true): -daystart -depth -follow --help
      -maxdepth LEVELS -mindepth LEVELS -mount -noleaf --version -xdev
tests (N can be +N or -N or N): -amin N -anewer FILE -atime N -cmin N
      -cnewer FILE -ctime N -empty -false -fstype TYPE -gid N -group NAME
      -ilname PATTERN -iname PATTERN -inum N -ipath PATTERN -iregex PATTERN
      -links N -lname PATTERN -mmin N -mtime N -name PATTERN -newer FILE
      -nouser -nogroup -path PATTERN -perm [+-]MODE -regex PATTERN
      -size N[bckw] -true -type [bcdpfls] -uid N -used N -user NAME
      -xtype [bcdpfls]
actions: -exec COMMAND ; -fprint FILE -fprint0 FILE -fprintf FILE FORMAT
      -ok COMMAND ; -print -print0 -printf FORMAT -prune -ls
```

Für den normalen Einsatz ist es allerdings ausreichend, wenn Sie sich folgendes Beispiel einprägen:

```
bash-2.02$ find / -name resolv.conf
/etc/resolv.conf
find: /var/spool/cron/atjobs: Permission denied
find: /var/spool/cron/atspool: Permission denied
find: /var/lib/xdm/authdir: Permission denied
```

Nach einiger Zeit hat `find` die Datei im Verzeichnis `/etc/` gefunden. Verzeichnisse, auf die `find` keinen Zugriff hat, werden als Fehlermeldung ausgegeben. Direkt hinter dem Kommando `find` können Sie das Verzeichnis angeben, in dem mit der Suche begonnen werden soll. Im Beispiel wird das gesamte Dateisystem (`/`) durchsucht. Wenn Sie den Namen einer Datei nicht genau kennen, können Sie auch nur einen Teil des Namens angeben und den Rest mit den üblichen Wildcards ersetzen. Beachten Sie, daß jedes Kommando zuerst von der Shell interpretiert und dann ausgeführt wird. Sie müssen also beispielsweise den `*` vor der Shell „verstecken“, indem Sie das Zeichen `\` voranstellen. Die Shell wird nun das folgende Zeichen ignorieren und direkt an das Kommando weiterreichen:

```
bash-2.02$ find / -name resol\*
/etc/resolv.conf
find: /var/spool/cron/atjobs: Permission denied
find: /var/spool/cron/atspool: Permission denied
find: /var/lib/xdm/authdir: Permission denied
```

Der zweite Weg, um Dateien zu finden, bietet sich über das Programm `locate`. Dieses ist um einiges schneller beim Finden von Dateien, da es eine Datenbank benutzt, die einmal am Tag aktualisiert wird. Es ist also nicht notwendig, jedesmal die komplette Platte zu durchsuchen. Allerdings funktioniert das Aktualisieren der Datenbank nur, wenn Ihr Rechner zu der Zeit in Betrieb ist, zu der auch diese Aktualisierung stattfindet. Da hierbei die komplette Festplatte durchsucht wird, kann der Vorgang einige Zeit dauern.

Sollten Sie Ihren Rechner ausschalten, so können Sie auch (als Superuser) die Datenbank von `locate` – mit dem Kommando `updatedb` zu jeder anderen Zeit aktualisieren. Sie können aber auch die Zeit, zu der `updatedb` gestartet wird, in der Datei `/etc/crontab` Ihren Bedürfnissen anpassen.

Wenn Ihnen diese Änderungen zu kompliziert erscheinen, können Sie auch das Paket `anacron` installieren, dieses sorgt dafür, daß Cronjobs, die eigentlich während der Zeit ausgeführt werden sollten, zu der Sie Ihren Rechner ausgeschaltet hatten, nachträglich ausgeführt werden.

Dadurch, daß die Datenbank einmal täglich aktualisiert wird, kann `locate` natürlich auch nur Dateien finden, die zu diesem Zeitpunkt bereits vorhanden waren. Später erzeugte Dateien werden von `locate` nicht angezeigt. Das klang jetzt etwas aufwendig, `locate` ist aber sehr leicht zu bedienen, wie folgendes Beispiel beweist:

```
bash-2.02$ locate XF86Config
/etc/X11/XF86Config
/usr/X11R6/lib/X11/XF86Config
/usr/X11R6/lib/X11/XF86Config.eg
/usr/X11R6/man/man5/XF86Config.5x.gz
```

Wie Sie sehen, findet `locate` alle Dateien, die den Suchbegriff beinhalten, ohne daß Sie mit Wildcards arbeiten müssen.

Wenn Sie `find` und `locate` vergleichen, werden Sie feststellen, daß `find` leistungsfähiger ist, `locate` dagegen im täglichen Gebrauch Dateien wesentlich schneller finden kann.

Wenn Sie häufiger mit `locate` oder auch die Datenbank nicht als Superuser aktualisieren lassen, werden Sie merken, daß unter Umständen nicht alle Dateien angezeigt werden. Dies liegt daran, daß nur Dateien in die Datenbank wandern, auf die das Programm `updatedb` Zugriff hat.

Weiterhin zeigt `locate` aber auch Dateien an, die ein normaler User nicht sehen sollte, dies ist vielleicht nicht gewünscht.

Debian GNU/Linux beinhaltet noch das Paket `slocate`. Diese spezielle Version zeigt nur die Dateien an, auf die der jeweilige User auch Zugriff hat. Sie können zusätzlich das Paket `suidmanager` installieren, damit kann `slocate` über das dazugehörige Programm `suidregister` Ihnen auch die Dateien anzeigen, auf die Sie normalerweise keinen Zugriff haben, natürlich erst nach Angabe des entsprechenden Paßwortes.

Beachten Sie bitte auch den Hinweis am Ende der Installation von `slocate`:

```
sushi:/root# apt-get install slocate
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  slocate
0 packages upgraded, 1 newly installed, 0 to remove and 27 not upgraded.
Need to get 0B/23.2kB of archives. After unpacking 143kB will be used.
Selecting previously deselected package slocate.
(Reading database ... 67742 files and directories currently installed.)
Unpacking slocate (from .../utils/slocate_2.1-5.1.deb) ...
Adding ‘diversion of /usr/bin/locate to /usr/bin/locate.notslocate by slocate’
```

```
Adding ‘diversion of /usr/bin/updatedb to /usr/bin/updatedb.notslocate by slocate’
Adding ‘diversi-
on of /usr/share/man/man1/locate.1.gz to /usr/share/man/man1/locate.notslocate.1.gz by slocate’
Adding ‘diversi-
on of /usr/share/man/man1/updatedb.1.gz to /usr/share/man/man1/updatedb.notslocate.1.gz by slocate
Adding ‘diversi-
on of /etc/cron.daily/find to /etc/cron.daily/find.notslocate by slocate’
Setting up slocate (2.1-5.1) ...
Adding group slocate (104)...
Done.
Changing permissions on: /usr/bin/slocate
Changing permissions on: /var/lib/slocate to: 0750

WARNING: You should run ’/etc/cron.daily/slocate’ as root. loca-
te will not work properly until you do or until it is run by cron (it is daily).
```

3.13.12. `gzip` – gepackte Dateien

Manchmal ist es sinnvoll, große Dateien zu komprimieren, sei es um Plattenplatz zu sparen, sei es um die Downloadzeiten zu verkürzen. Das Programm der Wahl unter Debian GNU/Linux ist `gzip` (GNU Zip).

Erstellen Sie zuerst eine Testdatei, um mit `gzip` experimentieren zu können, und sehen Sie sich die Größe dieser Datei an.

```
bash-2.03$ cd
bash-2.03$ cp /etc/profile ./testdatei
bash-2.03$ ls -l testdatei
-rw-r--r--    1 fr       fr            359 Jan 20 20:10 testdatei
```

Komprimieren Sie nun die Datei `testdatei` mit `gzip` und sehen Sie sich wieder das Ergebnis an:

```
bash-2.03$ gzip testdatei
bash-2.03$ ls -l testdatei.gz
-rw-r--r--    1 fr       fr            275 Jan 20 20:10 testdatei.gz
```

Beachten Sie, daß die Datei nun die Endung `.gz` bekommen hat. Somit ist klar zu erkennen, mit welchem Programm die Datei gepackt wurde und daß diese Datei überhaupt gepackt ist.

Um nun diese Datei wieder in den ursprünglichen Zustand zu versetzen, können Sie `gzip` mit der Option `-d` aufrufen:

```
bash-2.03$ gzip -d testdatei.gz
bash-2.03$ ls -l testdatei
-rw-r--r--    1 fr       fr            359 Jan 20 20:10 testdatei
```

Somit ist der alte Zustand wieder hergestellt. Der Erfolg ist bei so kleinen Dateien nicht sehr groß, Sie können das gleiche Experiment aber auch mit anderen, größeren Dateien probieren.

3.13.13. `split` – geteilte Dateien

Sicher standen Sie schon einmal vor dem Problem, daß eine Datei zu groß war. Sei es, um diese auf einem Medium zu transportieren oder um diese übers Netz zu verschicken. Unter Debian GNU/Linux ist es mit den Programmen `split` und `cat` möglich, Dateien zu zerteilen und wieder zusammenzufügen.

Kopieren Sie die Datei `/bin/bash` in Ihr Homeverzeichnis. Diese Datei hat eine Größe von etwas mehr als 450 Kilobyte.

```
bash-2.03$ cd
bash-2.03$ cp /bin/bash .
bash-2.03$ ls -l bash
-rwxr-xr-x    1 fr       fr         461720 Jan 22 15:42 bash
```

Sie können nun diese Datei mit dem Kommando `split` in kleinere Stücke teilen. Hierzu benötigt `split` Angaben über die maximale Größe der einzelnen Dateien sowie die Erweiterung des Dateinamens, die an jede Datei angehängt werden soll.

Mit der Option `-b` teilen Sie `split` die Größe mit. Ohne weitere Angaben geht `split` davon aus, daß der Wert in Byte angegeben wurde. Das ist natürlich nicht sehr praktikabel. Deshalb können Sie hinter dem Zahlenwert die Buchstaben `k` für Kilobyte oder `m` für Megabyte angeben.

Als Kennung für die einzelnen Dateien erweitert `split` den Dateinamen der ersten Datei mit `aa`, den zweiten mit `ab` und so weiter. Wenn Sie eine eigene Erweiterung (Prefix) zu jeder Datei erzeugen wollen, so können Sie diese mit angeben, hier im Beispiel wird „einzel“ angegeben:

```
bash-2.03$ split -b100k bash einzel
bash-2.03$ ls -l einzel*
-rw-r--r--    1 fr       fr         102400 Jan 22 15:42 einzelaa
-rw-r--r--    1 fr       fr         102400 Jan 22 15:42 einzelab
-rw-r--r--    1 fr       fr         102400 Jan 22 15:42 einzelac
-rw-r--r--    1 fr       fr         102400 Jan 22 15:42 einzelad
-rw-r--r--    1 fr       fr          52120 Jan 22 15:42 einzelae
```

Das Zusammenfügen der Dateien ist ebenfalls sehr einfach.

```
bash-2.03$ cat einzel* > bash-neu
bash-2.03$ ls -l bash*
-rwxr-xr-x    1 fr       fr         461720 Jan 22 15:42 bash
-rw-r--r--    1 fr       fr         461720 Jan 22 15:43 bash-neu
```

Anhand der Länge sehen wir, daß die Datei wieder hergestellt wurde.

3.13.14. `tar` – archivieren von Dateien

Häufig bekommt man Archive aus dem Netz in gepackter Form. Das Packen von Archiven beinhaltet zwei Dinge: erstens die Zusammenfassung von mehreren Dateien zu einer einzigen und zweitens das Komprimieren der Daten, um Plattenplatz zu sparen oder auch die Übertragungszeit zu verringern. Historisch gesehen verteilen sich diese beiden Funktionen unter Unix auch auf zwei Programme. Üblicherweise benutzt man zum Zusammenfassen der Dateien das Programm `tar` (Tape Archive) und zum Komprimieren das Programm `gzip`. Etwas bessere Ergebnisse beim Komprimieren von Daten erreicht das (neuere) Programm `bzip2`. Die GNU-Version des Programms `tar`, welche auch bei Debian GNU/Linux verwendet wird, kann beide Komprimierungsverfahren benutzen, so daß man nicht mit verschiedenen Programmen hantieren muß.

Altgediente Programme, wie zum Beispiel `tar`, verfügen häufig über eine Vielzahl von Funktionen. Einige davon werden heute kaum noch verwendet, sind aber trotzdem noch aus Kompatibilitätsgründen verfügbar, beispielsweise können Sie die Blockgröße bestimmen, mit der die Daten auf das Medium geschrieben werden. Eine solche Funktion werden Sie heute nur noch in sehr seltenen Fällen benötigen.

Einen Überblick über die Optionen von `tar` bekommen Sie mit der Option `--help`:

```
bash-2.03$ tar --help

GNU ‘tar’ saves many files together into a single tape or disk archive, and
can restore individual files from the archive.

Usage: tar [OPTION]... [FILE]...

Examples:
  tar -cf archive.tar foo bar  # Create archive.tar from files foo and bar.
  tar -tvf archive.tar         # List all files in archive.tar verbosely.
  tar -xf archive.tar          # Extract all files from archive.tar.

If a long option shows an argument as mandatory, then it is mandatory
for the equivalent short option also.  Similarly for optional arguments.

Main operation mode:
  -t, --list              list the contents of an archive
  -x, --extract, --get    extract files from an archive
  -c, --create            create a new archive
  -d, --diff, --compare   find differences between archive and file system
  -r, --append            append files to the end of an archive
  -u, --update            only append files newer than copy in archive
  -A, --catenate          append tar files to an archive
      --concatenate       same as -A
      --delete            delete from the archive (not on mag tapes!)

Operation modifiers:
  -W, --verify               attempt to verify the archive after writing it
      --remove-files         remove files after adding them to the archive
  -k, --keep-old-files       don’t replace existing files when extracting
      --overwrite            overwrite existing files when extracting
  -U, --unlink-first         remove each file prior to extracting over it
      --recursive-unlink     empty hierarchies prior to extracting directory
  -S, --sparse               handle sparse files efficiently
  -O, --to-stdout            extract files to standard output
  -G, --incremental          handle old GNU-format incremental backup
  -g, --listed-incremental=FILE
                             handle new GNU-format incremental backup
      --ignore-failed-read   do not exit with nonzero on unreadable files

Handling of file attributes:
      --owner=NAME             force NAME as owner for added files
      --group=NAME             force NAME as group for added files
      --mode=CHANGES           force (symbolic) mode CHANGES for added files
      --atime-preserve         don’t change access times on dumped files
  -m, --modification-time      don’t extract file modified time
      --same-owner             try extracting files with the same ownership
      --no-same-owner          extract files as yourself
```

```
      --numeric-owner          always use numbers for user/group names
  -p, --same-permissions       extract permissions information
      --no-same-permissions    do not extract permissions information
      --preserve-permissions   same as -p
  -s, --same-order             sort names to extract to match archive
      --preserve-order         same as -s
      --preserve               same as both -p and -s

Device selection and switching:
  -f, --file=ARCHIVE             use archive file or device ARCHIVE
      --force-local              archive file is local even if has a colon
      --rsh-command=COMMAND      use remote COMMAND instead of rsh
  -[0-7][lmh]                    specify drive and density
  -M, --multi-volume             create/list/extract multi-volume archive
  -L, --tape-length=NUM          change tape after writing NUM x 1024 bytes
  -F, --info-script=FILE         run script at end of each tape (implies -M)
      --new-volume-script=FILE   same as -F FILE
      --volno-file=FILE          use/update the volume number in FILE

Device blocking:
  -b, --blocking-factor=BLOCKS   BLOCKS x 512 bytes per record
      --record-size=SIZE         SIZE bytes per record, multiple of 512
  -i, --ignore-zeros             ignore zeroed blocks in archive (means EOF)
  -B, --read-full-records        reblock as we read (for 4.2BSD pipes)

Archive format selection:
  -V, --label=NAME                   create archive with volume name NAME
              PATTERN                at list/extract time, a globbing PATTERN
  -o, --old-archive, --portability   write a V7 format archive
      --posix                        write a POSIX format archive
  -I, --bzip2                        filter the archive through bzip2
  -z, --gzip, --ungzip               filter the archive through gzip
  -Z, --compress, --uncompress       filter the archive through compress
      --use-compress-program=PROG    filter through PROG (must accept -d)

Local file selection:
  -C, --directory=DIR          change to directory DIR
  -T, --files-from=NAME        get names to extract or create from file NAME
      --null                   -T reads null-terminated names, disable -C
      --exclude=PATTERN        exclude files, given as a globbing PATTERN
  -X, --exclude-from=FILE      exclude globbing patterns listed in FILE
  -P, --absolute-names         don't strip leading '/'s from file names
  -h, --dereference            dump instead the files symlinks point to
      --no-recursion           avoid descending automatically in directories
  -l, --one-file-system        stay in local file system when creating archive
  -K, --starting-file=NAME     begin at file NAME in the archive
  -N, --newer=DATE             only store files newer than DATE
      --newer-mtime            compare date and time when data changed only
      --after-date=DATE        same as -N
      --backup[=CONTROL]       backup before removal, choose version control
      --suffix=SUFFIX          backup before removal, override usual suffix

Informative output:
      --help            print this help, then exit
```

```
      --version         print tar program version number, then exit
  -v, --verbose         verbosely list files processed
      --checkpoint      print directory names while reading the archive
      --totals          print total bytes written while creating archive
  -R, --block-number    show block number within archive with each message
  -w, --interactive     ask for confirmation for every action
      --confirmation    same as -w

The backup suffix is '~', unless set with --suffix or SIMPLE_BACKUP_SUFFIX.
The version control may be set with --backup or VERSION_CONTROL, values are:

  t, numbered     make numbered backups
  nil, existing   numbered if numbered backups exist, simple otherwise
  never, simple   always make simple backups

GNU tar cannot read nor produce '--posix' archives.  If POSIXLY_CORRECT
is set in the environment, GNU extensions are disallowed with '--posix'.
Support for POSIX is only partially implemented, don't count on it yet.
ARCHIVE may be FILE, HOST:FILE or USER@HOST:FILE; and FILE may be a file
or a device.  *This* 'tar' defaults to '-f- -b20'.

Report bugs to <bug-tar@gnu.org>.
```

> **Tip:** Wenn die Anzeige eines längeren Textes nicht komplett auf dem Bildschirm erfolgen kann, können Sie den Text seitenweise ausgeben lassen, indem Sie die Ausgabe über eine „Pipe“ an das Programm `more` weiterreichen: `tar --help | more`

Für den täglichen Gebrauch kommen Sie aber mit maximal zehn von diesen vielen Optionen aus. Auch in diesem Abschnitt zeigen wir einige Beispiele aus der Praxis auf.

3.13.14.1. `tar` – packen von Dateien

Um mehrere Dateien in einem Archiv zusammenzufassen, benutzen Sie die Optionen `-cf` (create – erzeugen und file – Datei):

```
linux:/home/fr# tar -cf /tmp/test.tar /etc/
tar: Removing leading '/' from member names
```

Dies erzeugt eine neue Datei (`/tmp/test.tar`) mit allen Dateien aus dem Verzeichnis `/etc/`. `tar` entfernt automatisch das jedem Pfad vorangestellte `/`, bevor die Dateien in das Archiv aufgenommen werden. Dies verhindert, daß beim späteren Entpacken versehentlich Daten überschrieben werden.

3.13.14.2. `tar` – Entpacken von Dateien

Um die Daten wieder zu entpacken, benutzen Sie die Option `-x` (extract – entpacken). Beachten Sie, daß die Daten immer an der aktuellen Position im Dateisystem entpackt werden. Wenn Sie sich nicht sicher sind, erstellen Sie ein temporäres Arbeitsverzeichnis und verschieben Sie danach die Daten an die gewünschte Stelle:

```
linux:/home/fr# mkdir bla
linux:/home/fr# cd bla
```

```
linux:/home/fr/bla# tar -xf /tmp/test.tar
linux:/home/fr/bla# ls
etc
```

tar zum Plaudern bringen...: Benutzen Sie die zusätzliche Option `-v`, um den Vorgang des Packens oder Entpackens der Daten verfolgen zu können: `tar -xvf /tmp/test.tar`.

3.13.14.3. `tar` – Komprimieren der Archive

Bisher haben wir `tar` zum Zusammenfügen von Dateien benutzt, nun werden wir das Archiv zusätzlich noch komprimieren. Hierzu stehen bei GNU-Tar die Optionen `-z` für das Komprimieren mit `gzip` und `-I` zum komprimieren mit `bzip2` zur Verfügung. Die Benutzung ist ganz einfach, verwenden Sie `tar` wie oben gezeigt und fügen Sie beispielsweise die Option `-z` hinzu.

Jedes Zeichen zählt...: Zur Vereinfachnung können Sie den Strich `-` vor den Optionen einfach weglassen, `tar` versucht die erste Zeichenkette hinter dem Kommando als Optionen zu interpretieren: `tar xvf /tmp/test.tar`.

Hier ein Beispiel, wie Sie ein gepacktes Archiv erzeugen können:

```
linux:/home/fr# tar cvfz /tmp/test.tar.gz /etc/
```

Beachten Sie, daß es üblich ist, entweder wie hier gezeigt die Endung `.gz` anzuhängen, oder aber die Kurzform `.tgz` zu verwenden.

`tar` kann mit beiden Endungen „umgehen“, genaugenommen ist der Dateiname völlig egal. Die Endungen dienen nur der besseren Übersicht für den Benutzer.

3.13.14.4. `tar` – Benutzung von Bandlaufwerken / Streamer

Wenn Sie die Daten auf einem Tape-Streamer speichern wollen, können Sie die Daten auch direkt auf das Gerät speichern. Geben Sie dazu statt dem Dateinamen des Archives einfach das entsprechende Device an:

```
linux:/home/fr# tar cvfz /dev/st0 /etc/
```

Das Device `/dev/st0` wird von Streamern benutzt, die über einen SCSI-Anschluß verfügen. Wenn Sie die Daten von diesem Gerät wieder einlesen wollen, benutzen Sie dazu ebenfalls das entsprechende Device.

```
linux:/home/fr# tar xvfz /dev/st0
```

3.13.15. `file` – Dateitypen

Es ist üblich, Dateien so zu benennen, daß der Typ der Datei aus dem Namen erkenntlich ist. Programme sind nicht zwingend auf eine solche Endung angewiesen, diese dient nur der besseren Übersicht für den Benutzer (mit Ausnahme von Dateinamen, die fest in das Programm einkompiliert sind, wie zum Beispiel Namen von Konfigurationsdateien). Textdateien bekommen die Endung `.txt`, Perl-Programme die Endung `.pl` und Bilder die Endung `.jpeg` oder `.tiff` und so weiter. Trotzdem kann es passieren, daß Dateien entweder keine oder eine falsche Endung haben. Debian GNU/Linux stellt Ihnen das Programm `file` zur Verfügung, welches die meisten Dateitypen ermitteln kann.

Die Benutzung von `file` ist denkbar einfach: rufen Sie das Programm einfach zusammen mit dem Namen der zu bestimmenden Datei(en) auf, Sie können hier einen oder mehrere Dateinamen, auch mit Wildcards, angeben:

```
bash-2.03$ file /bin/bash /etc/passwd /etc/init.d/lpd
/bin/bash:       ELF 32-bit LSB executable, Intel 80386, version 1, dynamical-
ly linked (uses shared libs), stripped
/etc/passwd:     ASCII text
/etc/init.d/lpd: Bourne shell script text
```

`file` listet nun die verschiedenen Dateinamen mit den Dateitypen auf.

3.14. Einige `bash`-Funktionen

Wir möchten hier nur kurz auf die Debian GNU/Linux-Standard-Shell, die `bash`, eingehen. Weitere Informationen finden Sie in der Man-Page zur `bash`.

Eine der nützlichsten Funktionen der `bash` ist die Möglichkeit, Programm- und Dateinamen zu vervollständigen. Sie können dies zu jeder Zeit am Shell-Prompt ausprobieren: tippen Sie ein paar Zeichen eines Befehls und drücken Sie die TAB-Taste. Wenn die Eingabe bis zu der Stelle, an der Sie die TAB-Taste gedrückt haben, eindeutig war, es also kein zweites Programm gibt, das mit den gleichen Buchstaben beginnt, wird die Eingabe automatisch vervollständigt. Sollte es zwei oder mehr Möglichkeiten geben, so werden Sie lediglich einen „Piepton“ hören. Ein nochmaliges Drücken der TAB-Taste zeigt Ihnen alle möglichen Alternativen an. Probieren Sie es an einem Beispiel einmal aus: nehmen wir an, Sie möchten sich die Datei `/var/log/syslog` ansehen. Dazu benutzen Sie das Programm `more`. Geben Sie also `mo` ein und drücken Sie die TAB-Taste zweimal. Nun sollten Sie zumindest die Programme `mount` und `more` angezeigt bekommen.

Geben Sie nun so lange weitere Zeichen ein, bis die Eingabe eindeutig ist, in diesem Beispiel sollte also ein `r` ausreichen. Drücken Sie nun die TAB-Taste, und der Befehl wird vervollständigt. Probieren Sie dies noch einmal (als Superuser) mit dem Dateinamen `/var/log/syslog` aus. Probieren Sie immer nach einigen Zeichen mit der TAB-Taste die Eingabe zu vervollständigen.

Eine weitere recht nützliche Funktion ist das Wiederholen von bereits eingegebenen Kommandos. Sie können mit den Pfeiltasten (AUF und AB) durch die sogenannte „History“ der `bash` blättern und bereits ausgeführte Kommandos noch einmal ausführen oder auch gleich auf der Kommandozeile ändern.

3.15. Pipes

Bereits sehr früh wurde das Prinzip der „Pipes“, Röhren ist keine schlechte Übersetzung, integriert. Sie können etwas „hineinschieben“ und am anderen Ende kommt es wieder heraus. Wie bereits beschrieben, gibt es sehr viele kleine, spezialisierte Programme unter Unix, die mit speziellen Parametern aufgerufen werden können. Sinnvoll wäre eine Schnittstelle zwischen diesen Programmen, um Daten auszutauschen oder auch das Ergebnis

eines Programmlaufs in einem weiteren Programm aufzubereiten. Diese Schnittstelle ist in Form von „Pipes“ realisiert. Sicher haben Sie schon das Kommando `ls` benutzt, um sich die Dateien in einem Verzeichnis anzeigen zu lassen. Wenn Sie aber in einem Verzeichnis mit sehr vielen Dateien die Anzahl der Dateien ermitteln möchten, kann das Zählen leicht etwas umständlich werden. Um Zeichen, Wörter oder Zeilen in einer Datei zu zählen, gibt es aber das Kommando `wc` (Word Count). Ein Weg wäre also, die Ausgabe von `ls -l` in eine Datei zu schreiben und mittels `wc -l` die Zeilen zählen zu lassen. Der Umweg über eine Datei läßt sich aber mittels einer Pipe umgehen.

Unix benutzt hierfür das Zeichen `|` (Pipe). Verknüpfen Sie einfach die beiden Kommandos mittels dieses Zeichens zu einer Zeile: `ls -l | wc -l`, gibt Ihnen die Anzahl der Dateien im aktuellen Verzeichnis aus. Nur ein kleiner Haken an dieser Stelle: `ls` gibt als erste Zeile keinen Dateinamen aus, sondern eine Zeile, in der Informationen über das Verzeichnis aufgezeigt werden, Sie müssen also vom Ergebnis eine Zeile subtrahieren, um auf das genaue Ergebnis zu kommen.

Ein weiteres Beispiel für die Benutzung von Pipes werden wir im folgenden Abschnitt zu `ps` aufzeigen. Sie werden im Laufe der Zeit an vielen Stellen auf weitere Anwendungsfälle stoßen.

3.16. `ps` und `/proc`

Um auf einem Debian GNU/Linux-System die vielen gleichzeitig laufenden Programme im Zaum halten zu können, muß man sich natürlich auch einen Überblick über diese verschaffen. Dazu dient unter anderem das Programm `ps`. `ps` liest die benötigten Informationen unter Linux aus dem Verzeichnis `/proc` des Dateisystems. `/proc` ist nicht tatsächlich auf einer Festplattenpartition abgelegt, sondern wird vom Kernel ständig aktualisiert und in den Verzeichnisbaum eingeblendet. Somit hat der Benutzer auf einfachste Art und Weise Zugriff auf die Informationen.

Rufen Sie `ps` einfach einmal ohne weitere Parameter in einer shell auf. Dies sollte Ihnen in etwa folgendes Ergebnis anzeigen:

```
bash-2.02$ ps
PID TTY          TIME CMD
3522 pts/5    00:00:00 bash
3523 pts/5    00:00:00 ps
```

Ohne weitere Optionen zeigt `ps` die Prozesse der aktuellen Shell an. In der ersten Spalte sehen Sie die Prozeß-ID (PID), diese dient dazu, ein Programm im System eindeutig zu identifizieren. Zu diesem Zeitpunkt sind das die Shell selber sowie das Programm `ps`, welches ja gerade gestartet wurde. `ps` verfügt über eine Vielzahl von Optionen, mit denen Sie sich detailliertere Informationen zu den laufenden Programmen ansehen können.

Eine Kurzübersicht über die verfügbaren Optionen erhalten Sie, wie bei allen GNU-Programmen, über die Option `--help`.

```
bash-2.02$ ps --help
********* simple selection *********  ********* selection by list *********
-A all processes                      -C by command name
-N negate selection                   -G by real group ID (supports names)
-a all w/ tty except session leaders  -U by real user ID (supports names)
-d all except session leaders         -g by session leader OR by group name
-e all processes                      -p by process ID
T  all processes on this terminal     -s processes in the sessions given
a  all w/ tty, including other users  -t by tty
```

```
g  all, even group leaders!          -u by effective user ID (supports names)
r  only running processes            U  processes for specified users
x  processes w/o controlling ttys    t  by tty
*********** output format **********  *********** long options ***********
-o,o user-defined  -f full           --Group --User --pid --cols
-j,j job control   s  signal         --group --user --sid --rows
-O,O preloaded -o  v  virtual memory --cumulative --format --deselect
-l,l long          u  user-oriented  --sort --tty --forest --version
                   X  registers      --heading --no-heading
********* misc options *********
-V,V show version       L  list format codes  f  ASCII art forest
-m,m show threads       S  children in sum    -y change -l format
-n,N set namelist file  c  true command name  n  numeric WCHAN,UID
-w,w wide output        e  show environment   -H process heirarchy
```

Am häufigsten werden Sie sicher Optionen wie `a`, `u`, `x` und `w` benutzen. Diese werden nach dem Kommando einfach zusammengefaßt, also beispielsweise `ps auxw`.

Nun noch das versprochene Beispiel zum Thema Pipe: Nehmen wir an, Sie benötigen die Prozeß-ID eines Programms, um es mittels `kill` zu beenden. Auf einem System mit vielen Prozessen kann dies ein Problem sein. Die Lösung ist eine Kombination aus den Programmen `ps` und `grep`, welche mittels einer Pipe verkettet werden. Das benötigte Kommando würde wie folgt aussehen: `ps aux|grep netscape`. `ps` mit den Optionen `aux` gibt alle laufenden Prozesse in einer ausführlichen Form aus. `grep` filtert aus der Ausgabe des Programms `ps` alle Zeilen heraus, in denen die Zeichenfolge `netscape` vorkommt. Sie sollten nun eine recht knappe Ausgabe bekommen und die gewünschte Prozeß-ID leicht finden können.

Weitere Informationen zu `ps` finden Sie in der Man-Page zu `ps` (`man ps`).

3.17. Links

Unter Unix werden sogenannte Links bereits seit vielen Jahren eingesetzt. Sie kennen diese vielleicht schon von anderen Betriebssystemen unter dem Namen „Verknüpfungen“. Um einen Link zu erzeugen, bedient man sich des Kommandos `ln`. Auch dieses Kommando verfügt über die Option `--help`, welche Ihnen eine kurze Information zu den verfügbaren Optionen gibt:

```
bash-2.02$ ln --help
Usage: ln [OPTION]... TARGET [LINK_NAME]
  or:  ln [OPTION]... TARGET... DIRECTORY
Create a link to the specified TARGET with optional LINK_NAME.  If there is
more than one TARGET, the last argument must be a directory;  create links
in DIRECTORY to each TARGET.  Create hard links by default, symbolic links
with --symbolic.  When creating hard links, each TARGET must exist.

  -b, --backup               make a backup of each existing destination file
  -d, -F, --directory        hard link directories (super-user only)
  -f, --force                remove existing destination files
  -n, --no-dereference       treat destination that is a symlink to a
                               directory as if it were a normal file
  -i, --interactive          prompt whether to remove destinations
  -s, --symbolic             make symbolic links instead of hard links
  -S, --suffix=SUFFIX        override the usual backup suffix
  -v, --verbose              print name of each file before linking
```

```
  -V, --version-control=WORD  override the usual version control
      --help                  display this help and exit
      --version               output version information and exit

The backup suffix is ~, unless set with SIMPLE_BACKUP_SUFFIX.  The
version control may be set with VERSION_CONTROL, values are:

  t, numbered     make numbered backups
  nil, existing   numbered if numbered backups exist, simple otherwise
  never, simple   always make simple backups

Report bugs to <bug-fileutils@gnu.org>.
```

Aber das sieht komplizierter aus als, es ist. In 98% aller Fälle wird ihnen `ln` in der „Sparversion“ als `ln -s originaldatei link-zur-datei` ausreichen. Weitere Informationen finden Sie wie immer auch in der Man-Page.

3.18. `vi`

Der Unix-Standard-Editor `vi` ist nach der Installation des Basissystems auf jedem Debian GNU/Linux-System verfügbar. `vi` wird schon seit vielen Jahren auf Unix-Systemen eingesetzt, seine für Anfänger kryptische Bedienung rührt aus der langen Geschichte dieses Editors her. In der Urzeit der Computertechnik standen keine aufwendigen grafischen Arbeitsplätze zur Verfügung. Textdrucker mit Tastatur oder – etwas moderner – Text-Terminals (VT100, ein Modell der Firma DEC, ist noch heute ein Begriff), die seriell an der Rechner angeschlossen wurden, waren Stand der Technik.

Trotzdem ist es sinnvoll, ein paar wenige Grundlagen über den `vi` zu erfahren. Dieser Editor ist einfach immer verfügbar, auch auf einem minimalem System. Wenn Sie sich etwas in den `vi` eingearbeitet haben, werfen Sie mal einen Blick auf `vim`, vi improved, der eine erweiterte Version des `vi` ist.

Beim `vi` wird zwischen einem Kommando- und einem Eingabemodus unterschieden. Durch Drücken der Taste `i` für „Input“ kommen Sie in den Eingabemodus. Ein Druck auf die Taste `Esc` beendet den Eingabemodus und man befindet sich wieder im Kommandomodus. Zum Eingabemodus gibt es nicht viel zu sagen, es können damit weitere Zeichen in die Datei eingegeben werden.

Interessanter ist der Kommando-Modus des `vi`. Mit einzelnen Tasten können Sie im Text navigieren oder auch Zeichen/Zeilen löschen. Laden Sie einfach eine Datei, beispielsweise mit dem Kommando: `cp /etc/hosts .` (legt eine Kopie der Datei im aktuellen Verzeichnis ab), `vi ./hosts` (startet `vi` und lädt die Datei). In dieser Datei befinden sich die lokalen Zuordnungen des Rechnernamens zur IP-Nummer. Nach dem Start des vi befinden Sie sich im Kommandomodus. Sie können nun ein einzelnes Zeichen löschen, indem Sie die Taste `x` drücken. Ein solcher Befehl kann mit der Taste `u` rückgängig gemacht werden.

Um den Cursor im Text zu bewegen, können Sie die Pfeiltasten benutzen. Sollte dies aufgrund fehlerhafter Einstellungen (zum Beispiel über eine telnet-Verbindung) einmal nicht funktionieren, können Sie in jedem Fall mit den Tasten `h` (links), `j` (runter), `k` (rauf) und `l` (rechts) navigieren.

Viele der `vi`-Kommandos lassen sich auch „vervielfältigen“. Beispielsweise löscht `9x` 9 Zeichen ab der aktuellen Position. Dies funktioniert mit den meisten anderen Kommandos ebenso.

Wenn Sie die Änderungen in einer Datei speichern wollen, können Sie dies mit `:w` tun. Dateien unter einem anderen Namen speichern Sie mit `:w neu.txt`. Sie können den `vi` beenden, indem Sie `:q` eingeben. Auch hier sind Kombinationen möglich, so können Sie eine Datei mit `:wq` speichern und den Editor verlassen.

Häufig möchte man Zeilen kopieren: hierzu dient der Befehl `yy`. Dieser speichert die aktuelle Zeile in einem Puffer, die gespeicherten Daten lassen sich mit `p` wieder an einer anderen Stelle einfügen. Analog dazu lassen sich mit `7yy` 7 Zeilen kopieren... und so weiter.

Soweit zu den Grundzügen des Editors `vi`. Mit diesen wenigen Kommandos sind Sie in der Lage, Anpassungen an den Konfigurationsdateien Ihres Debian GNU/Linux-Systems vorzunehmen.

3.19. Dateisysteme

Als Dateisystem bezeichnet man auf einem Debian GNU/Linux (und auf allen anderen Unix-Systemen) den kompletten Verzeichnisbaum ab dem „root-“Verzeichnis `/`. Als Dateisystem wird aber auch die Organisationsform der Daten auf einem Medium (Festplatte, Diskette) bezeichnet, die von Betriebssystem zu Betriebssystem unterschiedlich ist.

Jedes physikalische Gerät, auf dem Sie Daten speichern wollen, müssen Sie zunächst mit einem Dateisystem versehen. Wenn Sie verschiedene Partitionen auf einem Medium erstellen, kann jede dieser Partitionen mit einem anderen Dateisystemtyp versehen werden. Jedes Betriebssystem verwendet mindestens einen eigenen Dateisystemtyp, viele verwenden auch mehrere oder können mit verschiedenen Typen umgehen.

Häufig sind im Debian GNU/Linux-Umfeld Kombinationen aus Linux und Windows-Dateisystemen anzutreffen. Debian GNU/Linux kann mit einer großen Zahl von Dateisystemen umgehen, Sie können somit sehr einfach Ihre Daten von anderen Betriebssystemen auf Ihr Debian GNU/Linux-System übernehmen.

3.19.1. `cfdisk` und `mount` – Einbinden eines Dateisystems

Unter Debian GNU/Linux gibt nur einen einzigen Verzeichnisbaum (beginnend mit `/`). In diesem sind, als Unterverzeichnisse, alle physikalischen Geräte zu finden. Es werden keine Buchstaben zu Identifikation der Geräte benutzt.

Mit Ausnahme des Root-Dateisystems (`/`, welches beim Systemstart automatisch angemeldet wird) müssen Sie alle weiteren Dateisysteme erst einmal in das System einbinden. Dabei kann, wie bereits beschrieben, jedes physikalische Gerät über mehrere Partitionen verfügen. Jedes dieser Dateisysteme wird im System in einem Verzeichnis (dem sogenannten „mount-point“) „eingehängt“. Das ist so einfach, wie es sich anhört, Sie können einen Mountpoint mit dem Kommando `mkdir` erzeugen und an dieser Stelle im Dateisystem das Medium einhängen.

Um dem System weitere Dateisysteme hinzuzufügen (dies können Sie im Normalfall nur als Superuser tun), dient das Kommando `mount`. Bei der Installation von Debian GNU/Linux von CD-ROM wurde das Dateisystem (`/cdrom`) bereits benutzt und vom Installationsprogramm ins System eingebunden. Hierbei wurde ein Link von dem entsprechenden Device auf das neue Device: `/dev/cdrom` angelegt. Sie müssen sich somit nicht das Device Ihres CD-ROMs merken (oder herausfinden), sondern können diesen Link benutzen. Weiterhin wurde bei der Installation das Verzeichnis `/cdrom` angelegt. Sie können nun eine eingelegte CD sehr leicht mit dem Kommando `mount /dev/cdrom /cdrom` „mounten“.

Der Mount-Point kann anstelle von `/cdrom` jedes andere, beliebige Verzeichnis sein. Beachten Sie jedoch, daß Verzeichnisse, an deren Stelle Sie ein Dateisystem mounten möchten, keine weiteren Dateien enthalten sollten. Das Mounten eines Dateisystems funktioniert auch, wenn sich bereits Dateien in dem Verzeichnis befinden, Sie können lediglich nicht mehr auf diese Dateien zugreifen. Die Dateien werden nicht gelöscht, sie werden praktisch von dem gemounteten Dateisystem „überlagert“.

In der Praxis reicht dieses Wissen jedoch nicht lange aus. Sicher werden Sie irgendwann den Wunsch haben, den Festplattenplatz Ihres Systems zu erweitern. Der erste Schritt ist natürlich der mechanische Einbau der Festplatte. Schon hierbei (eigentlich schon beim Kauf der Platte) müssen Sie sich für den Anschluß an einem der beiden IDE-Busse oder am SCSI-Hostadapter entscheiden.

Bei einem IDE-System notieren Sie sich, ob Sie die Platte am ersten (primary) oder am zweiten (secondary) Bus und ob die Platte als erste (Master) oder zweite (Slave) am jeweiligen Bus betrieben wird.

Am SCSI-Bus notieren Sie sich die ID der Platte und kontrollieren ob, und wenn ja mit welcher ID, noch weitere Geräte angeschlossen sind. Beachten Sie hierbei auch externe Geräte!

Im nächsten Schritt müssen Sie mindestens eine Partition auf der neuen Platte anlegen. Diese kann den gesamten Festplattenplatz in Anspruch nehmen, Sie können aber auch mehrere, kleine Partitionen anlegen. Unter Debian GNU/Linux haben Sie die Auswahl zwischen zwei Programmen: `fdisk` und `cfdisk`. Wir werden im folgenden `cfdisk`, aufgrund der ansprechenderen Oberfläche, vorstellen.

Ermitteln Sie zunächst das entsprechende Device für die neue Festplatte, mit Hilfe der zuvor notierten Daten. Für IDE-Laufwerke ist die Bezeichnung folgende:

- `/dev/hda` - Master am primären Bus
- `/dev/hdb` - Slave am primären Bus
- `/dev/hdc` - Master am sekundären Bus
- `/dev/hdd` - Slave am sekundären Bus

Dabei ist unerheblich, ob es sich um eine Festplatte oder ein CD-ROM handelt.

Bei SCSI-Laufwerken ist die Bestimmung etwas anders. Zunächst ist zu beachten, daß zwischen Festplatten, CD-ROMs/CD-Brennern und anderen Geräten unterschieden wird. Die Gerätedateien für SCSI-Festplatten werden mit `/dev/sdX` bezeichnet, wobei `X` für einen Buchstaben steht, angefangen bei `a` und dann aufsteigend nach SCSI-ID zugeordnet. Hier ein denkbares Beispiel:

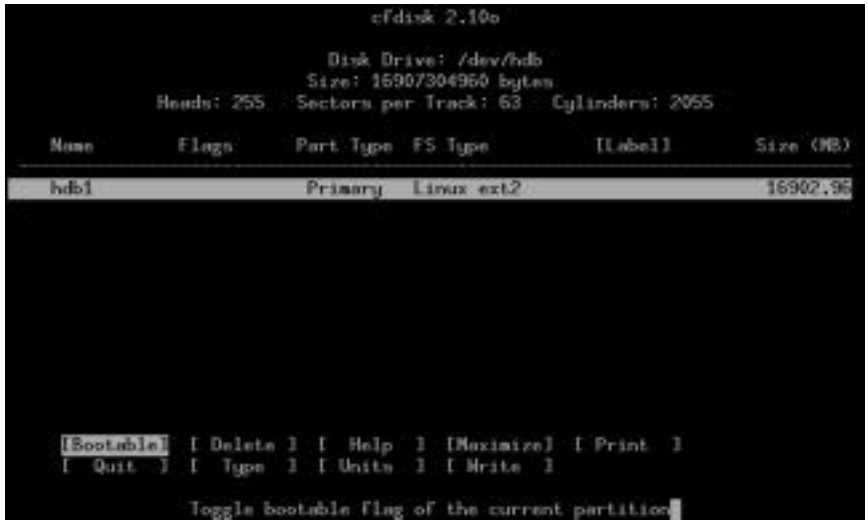
- `/dev/sda` – SCSI-Platte mit der kleinsten ID (z.B.: `0`)
- `/dev/sdb` – SCSI-Platte mit der mittleren ID (z.B.: `2`)
- `/dev/sdc` – SCSI-Platte mit der größten ID (z.B.: `3`)

SCSI CD-ROMs oder CD-Brenner werden ähnlich bezeichnet. Die Gerätedatei wird als `/dev/scdX` bezeichnet, hier steht `X` für eine Zahl, beginnend bei `0`. Beispielsweise:

- `/dev/scd0` - SCSI CD-ROM/Brenner (z.B. ID: `3`)
- `/dev/scd1` - SCSI CD-ROM/Brenner (z.B. ID: `5`)

Natürlich müssen/können Sie auf einer CD-ROM kein Dateisystem anlegen, dies sollte hier nur der Anschauung dienen.

Nachdem Sie nun das Ihrer Platte entsprechende Device bestimmt haben, können Sie `cfdisk` mit dem entsprechenden Device als Option starten, beispielsweise: `cfdisk /dev/hdb`. Bei einer neuen Festplatte werden nach dem Programmstart keinerlei Partitionen angezeigt. In diesem Beispiel, mit einer 16GB-Festplatte, wurde eine einzige Partition mit einem Linux Extended 2 (ext2-) Dateisystem angelegt.



Die erste Partition wird als `/dev/hdb1` angelegt, eine zweite würde `/dev/hdb2` genannt werden und so weiter...

Sie können sich innerhalb von `cfdisk` mit den Cursortasten bewegen und mit der RETURN-Taste den ausgewählten Menüpunkt auswählen.

Wenn Sie alle gewünschten Partitionen, oder auch nur eine einzige, angelegt haben, können Sie nun das eigentliche Dateisystem auf der Partition erzeugen. Hierzu steht Ihnen unter Debian GNU/Linux das Kommando `mke2fs` zu Verfügung. Auch diesem Kommando müssen Sie natürlich wieder angeben, welche Festplatte und vor allem auch welche Partition Sie mit dem Dateisystem beschreiben wollen. Für unser Beispiel beginnen wir mit der ersten Partition auf unserer Platte, also dem Device `/dev/hdb1`:

```
sushi:~# mke2fs /dev/hdc1
mke2fs 1.15, 18-Jul-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
128256 inodes, 256032 blocks
12801 blocks (5.00%) reserved for the super user
First data block=0
8 block groups
32768 blocks per group, 32768 fragments per group
16032 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Die bei Ihnen angezeigten Werte werden, je nach verwendeter Platte, von den hier gezeigten abweichen. (Gute Beobachter werden bemerken, daß in diesem Beispiel nicht die 16GB-Platte formatiert wurde, diese stand leider nicht zur Verfügung...)

Nun können Sie die frisch formatierte Partition zu Ihrem Dateisystem hinzufügen: `mount /dev/hdb1 /mnt` und mit dem Kommando `df` überprüfen, ob nichts schiefgelaufen ist.

Die Ausgabe könnte auf einem System mit mehreren Festplatten wie folgt aussehen:

```
sushi:~# df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda1              5767132   4352356   1121804  80% /
/dev/hdb1             16247612  14355348   1066928  94% /home/ftp
/dev/hdc1             24597980  21574360   1774080  93% /home/ftp/debian
/dev/hda2             18263244  15157492   2178016  88% /home/ftp/images
```

In dieser Auflistung sollten Sie dann auch die neu eingebundene Festplattenpartition finden.

3.19.2. `/etc/fstab` – Dateisysteme automatisch einbinden

Vielleicht werden Sie nach einem Neustart des Systems bemerkt haben, daß die neu eingebundene Festplatte nicht automatisch ins System eingebunden wird. Wenn Sie möchten, daß bestimmte andere Partitionen zusätzlich zu `/` (root) ins System eingebunden werden, so müssen Sie diese in die Datei `/etc/fstab` (für: „file system table“) aufnehmen. Weiterhin ist es sinnvoll, dort auch Dateisysteme einzutragen, die zwar nicht automatisch gemountet werden sollen, auf die Sie aber trotzdem schnellen Zugriff haben möchten, beispielsweise CD-ROMs, die öfter gewechselt werden.

Nach der Basisinstallation von Debian GNU/Linux sind bereits einige Einträge in der Datei `/etc/fstab` vorhanden:

```
# /etc/fstab: static file system information.
#
# <file system>     <mount point>   <type>  <options>   <dump >  <pass>
/dev/hda1            /               ext2    defaults    0       1
/dev/hda2            none            swap    sw          0       0
proc                 /proc           proc    defaults    0       0
```

Bei der Installation wurden (mindestens) das Root-Dateisystem (`/`) sowie eine Swap-Partition angelegt. Der dritte Eintrag dient dem „virtuellen“ Verzeichnis `/proc`, welches zur Laufzeit des Systems diverse Informationen zum System und zur Hardware enthält. Dieses verbraucht keinen Festplattenplatz.

Die erste Spalte beschreibt das Device und die zu mountende Partition. Die zweite Spalte verweist auf das Verzeichnis im Dateisystem, an der die Partition eingebunden werden soll. Beachten Sie, daß dieses Feld bei einer Swap-Partition mit dem Text „none“ anstatt eines Verzeichnisses gefüllt wird. Die dritte Spalte beschreibt den Typ des Dateisystems. Eine Beschreibung der weiteren Spalten finden Sie weiter unten, übernehmen Sie die Werte erst einmal wie gezeigt.

Um die im vorigen Abschnitt beschriebene Festplatte `/dev/hdc` automatisch ins System einzubinden, erweitern Sie die Datei um folgenden Eintrag:

```
/dev/hdc1            /mnt               ext2    defaults    0       2
```

Weitere nützliche Einträge sind folgende:

```
/dev/hdc             /cdrom          iso9660 ro          0       0
/dev/fd0             /floppy         auto    noauto,sync 0       0
```

Der erste Eintrag ermöglicht es Ihnen, eine CD-ROM einfach mit dem Kommando `mount /cdrom` anstatt `mount /dev/cdrom /cdrom` einzubinden. Gleiches gilt für die zweite Zeile, diesmal aber für das Diskettenlaufwerk. Um mit DOS-formatierten Disketten umzugehen, sehen Sie sich das Paket `mtools` an.

3.19.2.1. `/etc/fstab` – im Detail

Wie Sie schon gesehen haben, sind die Einträge in der Datei `/etc/fstab` in tabellarischer Form angeordnet:

```
# /etc/fstab: static file system information.
#
# <file system>     <mount point>   <type>  <options>   <dump >  <pass>
/dev/hda1            /               ext2    defaults    0       1
/dev/hda2            none            swap    sw          0       0
proc                 /proc           proc    defaults    0       0
```

Zeilen, die mit einem Kommentar (# - „Gartenzaun“) beginnen, können Sie ignorieren, das System tut dies auch.

Mit den ersten drei Spalten haben wir uns ja schon kurz beschäftigt, diese enthalten die Einträge für die Partition, den Mount-Punkt und den Dateisystemtyp. Die letzten drei Spalten bedürfen einiger Erklärungen.

Die fünfte Spalte wird von dem Programm `dump` benutzt um festzustellen, wann diese Partition gesichert werden soll. `dump` und `restore` dienen zur Sicherung von Daten.

Die sechste Spalte wird beim Systemstart von dem Programm `fsck` ausgewertet. Damit wird festgestellt, in welcher Reihenfolge die Dateisysteme beim Systemstart geprüft werden sollen. Die Root-Partition (`/`) sollte hier den Wert `1` erhalten. Dateisysteme, die nicht überprüft werden sollen, wie zum Beispiel `swap` oder CD-ROMs, bekommen den Wert `0`, alle anderen Dateisysteme bekommen eine `2`.

Nein, das ist kein Fehler im Text... zur vierten Spalte kommen wir jetzt. Diese bedarf einiger Erklärungen. Die Einträge in der vierten Spalte werden beim Mounten des Dateisystems benutzt. Sie können hier einen (im einfachsten Fall den Text: `default`) oder mehrere Werte angeben.

- `async` oder `sync` - Stellt die Datenübertragung (I/O) auf synchronen oder asynchronen Modus. Im synchronen Modus werden alle veränderten Daten sofort auf das Medium geschrieben, der asynchrone Modus speichert diese zwischen und schreibt später auf das Medium.
- `ro` oder `rw` - Mountet das Dateisystem zum „Nur-lesen“ (ro - read-only) oder zum Lesen und Schreiben (rw - read-write). Wenn Sie keine Änderungen an einem Dateisystem vornehmen wollen, so können sie dieses „ro“ mounten, um versehentliche Änderungen zu verhindern. Ebenso ist dieser Modus für CD-ROMs geeignet.
- `auto` oder `noauto` - Beim Systemstart oder wenn Sie das Kommando `mount -a` benutzen, werden alle Dateisysteme gemountet, welche Sie mit dem Eintrag `auto` versehen haben. Dateisysteme, die nicht dauerhaft zur Verfügung stehen, wie zum Beispiel Disketten oder CD-ROMs, sollten den Eintrag `noauto` bekommen. Sie verhindern so eine Fehlermeldung beim Systemstart, müssen diese Dateisysteme allerdings dann von Hand einbinden.
- `dev` oder `nodev` - `nodev` ignoriert die Gerätedateien auf dem gemounteten Dateisystem.
- `user` oder `nouser` - Normalerweise können Dateisysteme nur vom Superuser (root) in das System eingebunden werden. Mit dem Wert `user` erlauben Sie auch normalern Benutzern das Mounten von Dateisystemen. Sie können so beispielsweise den Zugriff auf das Diskettenlaufwerk oder das CD-ROM für alle Benutzer erlauben.
- `exec` oder `noexec` - Erlaubt oder verbietet das Ausführen von Programmen, die sich auf diesem Dateisystem befinden.
- `suid` oder `nosuid` - Wertet das Suid-Bit aus oder nicht.
- `defaults` - Der eigentlich wichtigste Wert, den dieses Feld annehmen kann. Aktiviert die Optionen: `rw, dev, suid, exec, auto, nouser, async`. Sie können einzelne Werte mit weiteren Parametern überschreiben.