Linux Primer

Eine begleitende Linuxeinführung zur Vorlesung
IT-Sicherheit an der h_da

Autor:
Timo Mühlbach

Version 1.0

11.04.2018

Contents

1 Was dieses Skript (nicht) ist

Dieses Skript ist in erster Linie für absolute Linux Neueinsteiger gedacht, die sich hier die für ITS[1] erforderlichen Kenntnisse über den Umgang mit Linux Betriebssystemen aneignen können. Dabei liefert es lediglich Hilfe zu Selbsthilfe und nicht etwa die vollständige Lösung Ihrer Praktikumsaufgaben. Ich habe versucht zu allen Themen qualitativ hochwertige weiterführende Links beizufügen, damit Sie tiefer in ausgewählte Themen einsteigen können. Dennoch werde ich nicht alle eventuellen Fragen abgedeckt haben - hierfür ist google immer noch Ihr bester Freund!

"Was hat IT Sicherheit mit Linux zu tun?" werden Sie sich eventuell jetzt fragen. Viele forensische Werkzeuge sowie Kryptologie- und Hashingprogramme sind in Linux frei verfügbar und durchaus Industriestandard.

Hinweis:
Oftmals sind insbesondere in Fußnoten gewisse Themen verlinkt. Es ist also empfohlen, dass sie dieses Skript in digitaler Form durcharbeiten!

"Suchst du eine helfende Hand? Du findest sie am Ende deines Armes."
— Arthur Lassen

[1] IT-Sicherheit

2 Konventionen in diesem Skript

Ich habe versucht, eine konsistente Verwendung der Formatierung des gesamten Skriptes beizubehalten, damit es sich so einfach und selbsterklärend liest, wie nur möglich, voraus gesetzt man kennt die verwendeten Konventionen:

Mit Strg werden Tasten oder Tastenkombinationen beschrieben, die Sie drücken müssen um die erläuterte Funktion auszulösen. Bei einzelnen Buchstaben wird immer der Großbuchstabe verwendet, damit die Anzeige eindeutig ist. In der Regel soll jedoch der Kleinbuchstabe verwendet werden, außer Shift ist teil der Tastenkombination oder es wird explizit im Text gefordert.

Kursiv geschriebener Text wird für Übersetzungen aus dem Englischen bzw. kurzen Erläuterungen von Abkürzungen etc. verwendet. Diese finden Sie häufig auch bei den erklärten Programmen und sollten sich als nützlich erweisen, um sich die Programmnamen merken zu können.

Fett gedruckte Wörter bzw. Passagen heben wichtige Merkmale hervor, die keinesfalls überlesen werden sollen, da sonst Missverständnisse entstehen können.

Mit `diesem Format` werden Kommandos angezeigt, die Sie in die Konsole eingeben sollen oder aus der Konsole ausgegeben werden. Aber auch einzelne Teile des Kommandos werden innerhalb der Erläuterung in diesem Format hervorgehoben um den Zusammenhang zu zeigen.

`Diese Monospaced` Schrift wird verwendet, um beispielsweise Ordner- oder Dateipfade anzuzeigen. Bei Mehrzeiligen Listings kann es sich aber auch ebenfalls um Eingaben handeln.

Hinweis:
Eingaben werden in der Regel ein `$␣` vorangestellt, ähnlich wie Sie es später bei Sich in der Konsole auch sehen. Dieses "Dollar Leerzeichen" sollen Sie selbstverständlich nicht mit eingeben.

Wenn sie einen Begriff oder Satz in Eckigen Klammern (z.B. `<Ihr Name>`) sehen, sollen Sie die Phrase durch etwas freies ersetzen und zwar ohne die Eckigen Klammern selbst. Also würde aus `<Ihr Name>` beispielsweise `Timo`.

Unklare Begriffe werden per Fußnote erläutert. Dort werden auch Links bereit gestellt.

3 Einleitung

Dieses Skript soll durch eine Mischung von theoretischen Grundlagen und praktischen Tipps und Tricks die Grundlagen beim Umgang mit *nix basierten Betriebssystemen vermitteln. Dazu gehören Linuxsysteme wie beispielsweise Ubuntu oder Debian aber auch MAC OSX.
Anlass für diese Einführung war die Erfahrung während meines ITS Moduls im WS14/15, dass viele meiner Kommilitonen Schwierigkeiten hatten, die für die Praktika geforderten Aufgaben effizient zu bearbeiten, da sie sich mit Linux ansich schwer taten.
Sie sollten sich darauf einstellen, nicht viele Screenshots in diesem Skript zu finden; der Grund dafür ist weniger meine Faulheit sondern viel mehr der Tatsache geschuldet, dass die Philosophie hinter Linux Programmierergemeinde lautet, dass ein Programm erst funktionieren soll, bevor es eine GUI[2] dazu gibt. Da die Entwicklung der Programme kontinuierlich voranschreitet, hinken existierende GUIs (- falls überhaupt vorhanden -) eigentlich immer in ihrer Funktionalität hinterher.

Ein weiterer Grund für die Bevorzugung von Kommandozeilenprogrammen ist die Tatsache, dass der geübte Anwender damit wesentlich schneller und somit effektiver arbeiten kann, als mit einer graphischen Oberfläche. Zugegeben: Bis man mit den gängigsten Befehlen vertraut ist, ist das Arbeitstempo natürlich langsamer, jedoch wird sich das bei regelmäßigem Üben schnell ändern. Falls Sie mir bei diesem Argument nicht ganz glauben wollen versuchen Sie einmal einen Tag lang am Computer jedes mal, wenn Sie etwas kopieren, ausschneiden oder einfügen möchten auf Strg + C / + X / + V zu verzichten und stattdessen das Kontextmenü über die rechte Maustaste zu verwenden. Sie werden schnell merken, wie viel Zeit es kostet, jedes mal die Hände von der Tastatur nehmen zu müssen.

Nachdem wir uns die grundlegende Strukturierung einer typischen Linuxinstallation angesehen haben, werden wir uns hauptsächlich mit dem richtigen und effizienten Umgang mit der Kommandozeile (im Folgenden auch Konsole, shell[3], bash[4], Terminal oder CLI[5] genannt) beschäftigen. Dabei werden wir uns auch ansehen, wie der Verzeichnisbaum strukturiert ist, wie die Benutzer- und Rechteverwaltung funktioniert, sowie wie man Dateien und Verzeichnisse anlegt, bewegt, kopiert, löscht, etc.

Sie sollten die Beispiele und Übungen in allen Kapiteln an der Bereitgestellten Kali VM selbst durcharbeiten. Fühlen Sie sich frei mit den gezeigten Befehlen zu experimentieren und herum zu spielen, damit Sie diese wirklich verstehen und verinnerlichen. Sie brauchen keine Angst haben irgendetwas (unwiederbringlich) kaputt zu machen - schließlich ist es eine virtuelle Maschine, die jederzeit auf ihren Ausgangszustand zurück gesetzt werden kann.

[2] *Graphical User Interface* - Grafische Benutzerschnittstelle
[3] Linux Konsole
[4] bourne again shell
[5] command line interface

4 Hinweise zur Virtuellen Maschine

Begleitend zum Skript und der IT Security Veranstaltung wird eine virtuelle Maschine bereit gestellt. Diese erhalten Sie in Form der Datei `KaliLinux.ova` zur Verfügung gestellt. Bitte installieren Sie sich zunächst (falls noch nicht vorhanden) *Oracle VM VirtualBox* auf Ihrem Rechner.
Um die virtuelle Maschine in Betrieb nehmen zu können, müssen Sie diese über das Menü `Datei -> Appliance importieren...` oder alternativ per Strg + I importieren.
Danach können Sie die Maschine über den Button `Starten` hochfahren und bekommen das laufende Betriebssystem in einem Fenster angezeigt.

In der virtuellen Maschine läuft Kali Linux, eine Penetration Testing Distribution, die alle für die Praktika benötigten Programme beinhaltet.
Es wurde sicher gestellt, dass alle Übungen und Erläuterungen innerhalb dieser Umgebung funktionieren.

Um sich nach dem Hochfahren in der virtuellen Maschine anzumelden, klicken Sie zunächst auf *Andere...*, dann geben Sie als Benutzer **root** und danach als Passwort **toor** ein. Jetzt sollten Sie den Desktop von Kali Linux sehen.

Oben in der Menü-Leiste des Desktop sehen Sie folgenden Button:  Mit diesem können Sie ein neues Terminal öffnen, in welchem die jeweiligen Übungen dann ausgeführt werden können.

5 Der erste Kontakt mit der Konsole - "Hello world!"

Um die lange Tradition des Hello world zu wahren, ist dies das Erste, was wir in der Konsole ausgeben wollen.
Die Konsole die in Linux standardmäßig läuft ist die `shell` bzw. `bash`[6]. Dies bezeichnet eine interpretierte Programmiersprache, die demnach keinen Kompilierungsschritt benötigt, sondern direkt ausgeführt werden kann. Gängige Aufgaben, die in der Shell erledigt werden ist das Ausführen von Programmen und eventuell die Ausgaben eines Programmes als Eingabe für das nächste Programm zu verwenden.[7]
Öffnen Sie nun ein Konsolen-fenster, tippen Sie Folgendes ein und führen Sie den Befehl per ENTER Taste aus:

Beispiel 5.1:
```
$ echo "Hello world!"
```

Wenn alles so funktioniert wie es sollte, sollten Sie nun `Hello world!` in der Zeile nach Ihrem Befehl sehen. Herzlichen Glückwunsch, Sie haben Ihren ersten shell Befehl ausgeführt! Schauen wir uns doch einmal genauer an, was hier passiert ist: Normalerweise besteht ein Befehl immer aus einer einzigen Zeile und startet mit dem eigentlichen Programm, welches ausgeführt werden soll. Alles Nachfolgende sind dann in der Regel eine durch Leerzeichen getrennte Liste an Argumente bzw. Optionen. (- Oft im auch Schalter oder Switches genannt. -) Der Unterschied zwischen Argumenten und Optionen besteht darin, dass Optionen eine Art Name haben, der sie Identifiziert, wohingegen Argumenten namenslos sind und allein durch Ihre Reihenfolge unterschieden werden können. Dazu später mehr. Wir können also `echo` als das Programm und `"Hello world!"` als ein Argument identifizieren. Eventuell Fragen Sie sich jetzt, warum `"Hello world!"` nur ein einziges Argument ist, obwohl die beiden Worte doch ganz klar durch ein Leerzeichen getrennt sind; die Erklärung hierfür ist, dass die beiden Worte in Anführungszeichen eingeschlossen sind. Die bash ist intelligent genug, Zeichenketten in Anführungszeichen oder Hochkommata als einzelnes Argument zu erkennen, ansonsten wäre es unmöglich die Zeichenkette `Hello World!` als ein Argument zu übergeben. Der Unterschied zwischen Anführungszeichen und Hochkommata wird im folgenden Abschnitt erläutert.

Übung 5.1:
Überlegen Sie sich, was das im Umkehrschluss für Zeichenketten bedeutet, die keine Leerzeichen beinhalten und wie diese als Argument übergeben werden können. Wie kann man folgenden Befehl vereinfachen?
```
$ echo "Hello"
```

Der Befehl `echo` macht nichts anderes als eine Zeichenkette, die es als erstes Argument erhält in der Konsole auszugeben.

[6] *bourne again shell* - Eine verbesserte Version der `shell`
[7] Für Interessierte: Das Prinzip folgt dem Design Pattern Pipes and Filter

5.1 Der Unterschied zwischen Argumenten und Optionen

Wie vorhin angesprochen gibt es neben Argumenten auch noch Optionen. Manchen Optionen können Werte zugewiesen werden, andere Optionen fungieren lediglich als Flag. Das bedeutet, dass die bloße Existenz dieser Option die Ausführung des Programmes verändert; daher auch die Bezeichnung Switch bzw. Schalter. In seltenen Fällen sind die Werte einer Option auch optional (z.B. weil es einen Standardwert gibt). Welche Optionen sowie welche Art von Optionen ein Programm hat, definiert das Programm selber. Wo Sie diese Information finden wird später im Skript erläutert. (Siehe Kapitel 9) Ein Vorteil der Optionen gegenüber der Argumente ist, dass Sie bei ihnen nicht auf eine Reihenfolge achten müssen, da sie sich über ihren Namen selbst identifizieren. Manche Optionen haben bei vielen Programmen auch Aliase: Eine Option, die in einem oder zwei Worten beschreibt, was sie tut, sowie eine Kurzform, die nur aus einem einzigen Buchstaben besteht, damit man die Kommandos schneller schreiben kann. Fast alle Optionen beginnen unter Linux mit einem einfachen oder doppeltem Bindestrich unmittelbar gefolgt von dem Namen der Option. Dabei wird der einfache Bindestrich für die Kurzform und ein doppelter Bindestrich in der Regel für die Langform verwendet. Falls Diese einen Wert akzeptiert folgt dieser nach einem Leerzeichen darauf. Es gibt jedoch auch (abhängig vom Programm) andere Varianten wie z.B. `-option=value` oder `-oValue` . Informieren Sie sich also im Zweifelsfall unbedingt über die gültige Syntax des jeweiligen Programms.

Ergänzen wir im Folgenden unser `echo` Beispiel nacheinander um zwei verschiedene Optionen:

Beispiel 5.2:

```
$ echo -e '\tHello world!'
$ echo -E '\tHello world!'
```

Sie sollten nun folgende Ausgaben auf Ihrer Konsole sehen:

```
    Hello world!
\tHello world!
```

Wie Sie erkennen handelt es sich bei den beiden Optionen um Wertelose Schalter. Das `-e` bzw. `-E` steht in diesem Fall für *Escape backslash*. Es ist erwähnenswert, dass dabei Kleinbuchstaben die positive Form und Großbuchstaben die Verneinung davon ist. `-e` steht demnach für *Do escape backslashes*, wohingegen `-E` für *Do not escape backslahes*, also: "Interpretiere keine Backslash Symbole" (d.h. Zeichenfolgen die mit einem Backslash beginnen und einem darauf folgenden Buchstaben, der angibt, um welches Symbol es sich handelt) steht. Damit ergibt sich ebenfalls, dass -e und -E gegensätzliche Schalter sind.

Übung 5.2:
Was passiert, wenn Sie den Befehl mit der Zeichenkette aus Beispiel 5.2 weder mit `-e` oder mit `-E` ausführen? Was lässt sich daraus schließen?

Der Unterschied zwischen Zeichenketten in Hochkommata und Anführungszeichen ist dabei, dass Backslashes in Anführungszeichen automatisch escaped werden, bevor sie als Argument an das Programm übergeben werden. So wird aus \\ automatisch \, selbst wenn bei dem `echo` Programm die `-E` Option gesetzt ist.

Übung 5.3:
Da mit `-e` Backslash Symbole interpretiert werden muss bei eingeschalteter Interpretation irgendwie dennoch ein Backslash auszugeben sein. Die Lösung dafür ist ein Backslash Symbol selbst: `\\` Überlegen Sie sich, mit welcher Zeichenkette Sie mit folgendem `echo` Befehl einen doppelten Backslash ausgeben lassen können. Führen Sie anschließend ihre Lösung aus und überprüfen Sie ihr Ergebnis:
`$ echo -E "<Ihre Lösung>"`

Wie können sie mit folgendem `echo` Befehl ebenfalls einen doppelten Backslash ausgeben lassen?:
`$ echo -e "<Ihre Lösung>"`

6 man pages

Wie bereits erwähnt, soll dieses Skript in erster Linie Hilfe zur Selbsthilfe bieten. man pages steht für *manual pages*, also die Anleitungsseiten der Programme. Dabei ist `man` das Programm, mit dem die Seiten angezeigt werden können. Führen Sie in der bash `man man` aus. Nun sollten Sie die Anleitung für man an sich sehen. Mit ↓ und ↑ bzw. Bild ↓ und Bild ↑ können Sie auf diesen Seiten runter bzw. hoch scrollen. Die man pages können Sie mit Q verlassen und nicht etwa mit ESC. Dies gilt im Übrigen für viele Programme, welche interaktiv in der bash laufen, wie etwa `less`, oder `vim`. Zu diesen Programmen später mehr. Innerhalb der man pages können Sie per `/Suchwort` nach bestimmten Wörtern suchen. Um zum nächsten Suchergebnis zu gelangen drücken Sie N, um zum vorherigen Suchergebnis zu gelangen drücken Sie ↑ + N. Um sich zusätzliche Steuerbefehle für die man pages anzusehen, drücken sie H. Bitte beachten Sie, dass ein Strg mit ^ angezeigt wird und anstatt SHIFT + X lediglich ein großes X statt einem kleinen x verwendet wird.

Übung 6.1:
Lesen Sie sich die man pages für `man` und `echo` durch.

Weitere Möglichkeiten Hilfe zu finden bieten die Programme `apropos` und `whatis`. Es gibt weitere Programme nach dem Schema `<engl. W-Fragewort>is`, wie z.B. `whereis`, welches zeigt, wo die Binärdateien für ein verfügbares Programm sind.

Übung 6.2:
Machen Sie sich mit den Programmen `apropos` und `whatis` vertraut und notieren Sie die Unterschiede zu `man`.

7 Effizienter Umgang mit bash

Da Sie nun wissen, was Programme mit ihren Argumenten und Optionen sind und wie Sie sich mit einem unbekannten Programm mittels der man pages vertraut machen können, wollen wir uns kurz den Möglichkeiten befassen, die die alltägliche Arbeit mit der bash vereinfachen. Über die bash können nämlich nicht nur Klartext Zeichen wie Buchstaben, Zahlen oder Sonderzeichen ein- und ausgegeben werden, sondern auch Tastenkürzel gesendet werden (ähnlich wie Strg + C um etwas zu kopieren). Gängige Tastenkürzel und deren Bedeutung in der bash sind:

Tastenkürzel	Erklärung
→ TAB	Vervollständigt einen Befehl, sofern Programm oder Pfad bekannt ist
Strg + R	Gefolgt von den ersten Zeichen des zuletzt verwendeten Befehls durchsucht in kürzlich eingegebenen Befehlen nach Übereinstimmungen um schnell einen Befehl zu vervollständigen
Strg + L	Leert die Terminalansicht
Strg + U	Schneidet alles **links** vom Cursor aus und speichert es in einer Terminal-internen Zwischenablage.
Strg + K	Schneidet alles **rechts** vom Cursor aus und speichert es in einer Terminal-internen Zwischenablage.
Strg + Y	Fügt den Inhalt der Zwischenablage ins Terminal ein.
Strg + C	Terminiert (beendet) ein im Vordergrund laufendes Programm im Terminal.

Insbesondere → TAB bietet eine massive Zeitersparnis, da Programme oder Datei- bzw. Ordnernamen sehr nach der Eingabe der ersten paar Buchstaben vervollständigt werden können und so nicht bis zum Ende ausgeschrieben werden müssen. Das ist auch nützlich, falls der genaue Name nicht bekannt ist.
Sollte die Vervollständigung beim ersten Druck auf → TAB nicht funktionieren kann dies drei Gründe haben:

1. Die bereits getippte Zeichenkette ist nicht eindeutig. Drücken Sie ein zweites mal → TAB um sich die möglichen Alternativen auflisten zu lassen.
2. Es gibt keinen Befehl und/oder Ordner, der mit der bereits getippten Zeichenkette beginnt. Ein zweites Antippen von → TAB hat ebenfalls keine Wirkung.
3. Sie versuchen Optionen / Argumente von Programmen zu vervollständigen und für dieses Programm ist keine Vervollständigung verfügbar[8]. Ein zweiter Druck auf → TAB hat hier ebenfalls keinen Effekt.

[8] Diese muss i.d.R. vom Programm mitgeliefert werden und funktioniert nicht etwa *magisch* für selbstgeschriebene Programme.

8 Der Linux Verzeichnisbaum

In Windows werden die verschiedenen logischen Laufwerke mit einem Großbuchstabe gefolgt von einem Doppelpunkt und Backslash (`C:\`) identifiziert. Ein **absoluter** Dateipfad startet also immer mit der Laufwerksbezeichnung gefolgt von beliebig vielen Ordnern und endet mit einem Dateinamen z.B. `C:\Ordner\Datei.txt`.

Bei Linux werden Ordner und Verzeichnisse nicht durch einen Backslash sondern durch einen (regulären) Forwardslash (im Folgenden nur noch Slash oder Schrägstrich) getrennt. Absolute Dateipfade starten auch nicht mit einer Laufwerksbezeichnung, sondern meist mit einem Slash und die verschiedenen Laufwerke sind in dem Verzeichnisbaum enthalten - wo wird später erläutert. Der äquivalente Dateipfad zum Beispiel oben lautet unter Linux also: `/Ordner/Datei.txt`.

Direkt im *Root Verzeichnis*[9] `/` befinden sich folgende Verzeichnisse mit deren Inhalt:

Verzeichnis	**Beschreibung**
/bin	Binärdateien (d.h. ausführbare Dateien)
/boot	Dateien, die zum Start (*booten* des Betriebssystems notwendig sind)
/dev	Gerätedateien (USB-Sticks, CD-/DVD-Laufwerke, etc.)
/etc	Konfigurationsdateien
/home	Homeverzeichnisse der Benutzer
/lib	Systembibliotheken
/media	Hier werden Laufwerke gemounted[10]
/mnt	Temporäre mounting points
/opt	*optionale* Anwendungsprogramme
/root	Das Homeverzeichnis des `root`-Benutzers
/sbin	*system binaries* - Ausführbare Systemdateien
/srv	Dateien für Systemdienste
/tmp	Temporäre Dateien
/usr	Eine "zweite" Dateiebene - kann in einer seperaten Partition angelegt und somit zwischen mehreren Rechnern geteilt werden
/var	Eine Art Arbeitsverzeichnis für Programme

Die Verzeichnisstruktur aller Linux Distributionen orientiert sich dabei an dem FHS[11], kann jedoch je nach gewählter Distribution abweichen.

[9] Wurzelverzeichnis, oberstes Verzeichnis
[10] Englisch für *Eingehängt*
[11] Filesystem Hierarchy Standard

9 Arbeiten mit Dateien

9.1 Durch das Dateisystem navigieren

Durch Verzeichnisse navigieren Sie mit dem Programm `cd` (*change directory*) mit dem gewünschten Zielpfad als 1. Argument. Dieser Zielpfad kann sowohl **absolut** als auch **relativ** sein. Dabei beginnt ein absoluter Pfad i.d.R. mit einem `/` gefolgt von den Ordnern. Bei einer relativen Angabe beginnt man hingegen den Pfad direkt mit dem Ordnernamen, der sich innerhalb des aktuellen Verzeichnisses befindet. Alternativ kann man eine relative Pfadangabe auch mit `./` beginnen. Diese Spezifikation ist besonders dann Sinnvoll, wenn man nicht das Verzeichnis wechseln möchte, sondern ein Programm im aktuellen Ordner ausführen möchte und sichergestellt sein muss, dass es kein gleichnamiges Programm im globalen Kontext existiert, welches ansonsten bevorzugt würde.

Eines der für Sie wichtigsten Verzeichnisse ist `/home/<Benutzername>`, dies ist ihr *Homeverzeichnis*, welches alleine Ihnen gehört (siehe Rechteverwaltung) und wo sich alle Ihre persönlichen Dateien befinden, vergleichbar mit `C:\Users\<Benutzername>` unter Windows. Dieses Verzeichnis kann auch per `~` (*Tilde*) angegeben werden. So bewirkt `cd ~` den Wechsel in das eigene Homeverzeichnis. Der Tilde kann auch direkt ein weiterer Unterpfad angehägt werden, z.B. `cd ~/Downloads`

Um heraus zu finden, in welchem Verzeichnis man sich aktuell befindet, kann man `pwd` (*print working directory*) ausführen. Dieses Programm gibt absolut den aktuellen Dateipfad aus.

9.2 Dateien und Verzeichnisse anzeigen

Um sich die Dateien und Verzeichnisse anzeigen zu lassen, verwenden Sie den Befehl `ls` (*list*). Sehen Sie sich die man pages des `ls` Befehls an und machen Sie sich insbesondere mit den möglichen Optionen vertraut. Eine gängige Kombination ist z.B. `ls -al` .

9.3 Dateien und Verzeichnisse manipulieren

Die vier Aktionen, welche auf eine Datei oder ein Verzeichnis angewendet werden können folgen (unabhängig von Linux) immer dem CRUD-Schema[12]

Create - Erstellen

Möchten Sie einen Ordner erstellen, so können Sie dies mit `mkdir` (*make directory*) erledigen; das Programm akzeptiert als erstes Argument den Ordnernamen

[12] Create, Read, Update, Delete - Erstellen, Lesen, Ändern, Löschen Der Begriff kommt aus dem Datenbankjargon, gilt jedoch auch für Dateioperationen

mit optional vorangestellter Pfadangabe. `mkdir foo` legt demnach einen Ordner namens *foo* im aktuellen Verzeichnis an, wohingegen `mkdir ~/foo` den Ordner *foo* als Unterordner Ihres Homeverzeichnisses erstellt.

Dateien können unter Linux entweder direkt über einen Editor (siehe *Update*) oder per `touch` angelegt werden. Das Programm `touch` akzeptiert genau wie `mkdir` als erstes Argument den Namen der anzulegenden Datei. Auch hier kann eine Pfadangabe dem Dateinamen vorangestellt werden.

Übung 9.1:
Legen Sie in Ihrem Homeverzeichnis einen Ordner namens *foo* an und danach eine Datei *testdatei.txt* in dem Ordner *foo* an.

Übung 9.2:
Unter Linux gibt es ebenfalls die Möglichkeit Dateilinks anzulegen. Dabei wird zwischen physikalischen und symbolischen Links unterschieden. Informieren Sie sich in den man pages oder online über die Unterschiede. Das Programm zum Anlegen von Dateilinks ist `ln` . Legen Sie danach einen symbolischen Link namens *testlink.txt* in Ihrem Homeverzeichnis an, welches auf `/foo/testdatei.txt` zeigt.

Read - Lesen

Um sich den Inhalt einer Klartextdatei ausgeben zu lassen gibt es mehrere Möglichkeiten je nach Anwendungsfall. Die einfachste Methode, sich eine Klartextdatei ausgeben zu lassen ist das Programm `cat <Dateiname>` . Dabei wird der Inhalt der Datei auf der Konsole von Anfang bis Ende ausgegeben. Dies kann beispielsweise bei sehr großen Dateien jedoch einige Probleme mit sich bringen. Um ein scrollbaren Ausschnitt der Datei von Anfang bis ende zu erhalten, können Sie `less <Dateiname>` benutzen. Innerhalb der Anzeige können Sie mit den Pfeiltasten hoch und runter scrollen. Verlassen wird die Ansicht mit Q . Möchten Sie eine Log-Datei ansehen, interessieren Sie in der Regel nur die letzten paar Zeilen. Da Logs typischerweise auch sehr groß sind, gibt es mit `tail <Dateiname>` die möglichkeit, sich nur ein Paar der letzten Zeilen ausgeben zu lassen. Zusätzlich können Sie mit der Option `-f` (*follow*) die Ausgabe automatisch aktualisieren lassen, falls neue Zeilen der Datei angehängt wird. Wenn Sie `tail` mit der Option `-f` verwenden, müssen Sie das Programm manuell mit Strg + C beenden.

Update - Ändern

Um eine Datei von einer Stelle zur Anderen zu kopieren, verwenden Sie üblicherweise
`cp <Quelldatei> <Zieldatei>`
(*copy*).
Wenn Sie sicher gehen müssen, dass die Datei Bit für Bit exakt identisch kopiert werden, benutzen Sie stattdessen
`dd if=<Quelldatei> of=<Zieldatei>` [13]
(*dump device*).

[13] Beispiel für Optionen, die nicht mit einem Bindestrich beginnen! Also **immer** RTFM ;)

Um eine Datei hingegen von einer Stelle zur Anderen zu verschieben, verwenden Sie `mv <Quelldatei> <Zieldatei>` (*move*).

Es gibt auch Texteditoren, die im Terminal laufen. Einer der bekanntesten Editoren ist Vim. Dieser kann per `vi` gestartet werden. Die Bedienung dieses Editors ist jedoch nicht trivial, deshalb wird auf eine Einführung in diesem Skript verzichtet. Interessierte finden online genug Informationen, um sich selber in Vim einzuarbeiten. Es gibt noch andere einfachere Möglichkeiten, die Ausgaben eines Programmes in eine Datei zu schreiben, diese werden später erläutert.

Delete - Löschen

Um eine Datei aus dem Dateisystem zu löschen, verwenden Sie den Befehl `rm <Dateiname>` (*remove*).
Wenn Sie versuchen einen Ordner zu löschen, bekommen Sie üblicherweise folgende Fehlermeldung zurück: `rm: Entfernen von „<Ordnername>“ nicht möglich: Ist ein Verzeichnis`
Um dennoch Verzeichnisse löschen zu können, müssen Sie lediglich den Schalter `-r` (*recursive*) hinzufügen. Damit werden rekursiv alle im Verzeichnis enthaltenen Dateien und Ordner gelöscht.

Übung 9.3:
Legen Sie in Ihrem `home`-Verzeichnis einen Ordner `uebungen` an. In diesem Ordner legen Sie eine Datei `datei1` an.
Danach kopieren Sie diese Datei in die Datei `datei2`.
Bewegen Sie anschließen die Datei `datei1` zu der Datei `datei3`
Sie sollten also nun in `/uebungen/` die beiden Dateien `datei2` und `datei3` haben.
Wechseln Sie zurück in Ihr `home`-Verzeichnis und legen Sie eine Datei `datei4` an.
Verschieben Sie diese anschließend in den Ordner `uebungen`.
Überprüfen Sie, ob alle drei Dateien `datei2`, `datei3` sowie `datei3` in dem Ordner vorhanden sind.
Anschließend löschen Sie den Ordner `uebungen`.

Eine weitere typische Dateioperation in der Linux Welt ist das Ausführen von Dateien. Darauf wird im nachfolgenden Kapitel näher eingegangen, wenn wir uns mit dem Rechtesystem von Linux beschäftigen.

10 Das Linux Rechtesystem

Das Linux Rechtesystem scheint auf den ersten Blick eventuell etwas aufgedunsen und unnötig Komplex. Auf den zweiten Blick werden Sie jedoch feststellen, dass es eigentlich relativ einfach und strukturiert und gleichzeitig sehr mächtig ist.

Zugrundeliegend sind vier Ideen:

1. Es gibt Benutzer und Gruppen. n Benutzer können zu m Gruppen gehören. Gruppen fassen gängige Aufgabenbereiche zusammen, wie z.B. Anwendung, Verwaltung oder Administration eines Systems.
2. **Jede** Datei und **jeder** Ordner gehört sowohl genau einem Benutzer als auch genau einer Gruppe.
3. Jeder Datei bzw. Jedem Ordner gehören genau drei Rechte-Tripel: Das Tripel für den Eigentümer (also den zugewiesenen Benutzer), das Tripel für die Gruppe und ein Tripel für alle Anderen (also jeden Benutzer, der nicht in der zugewiesenen Gruppe ist und auch nicht der zugewiesene Benutzer ist). Diese werden im Folgenden *Zielgruppe* genannt.
4. Ein Rechte-Tripel besteht für jede Zielgruppe aus denen im vorherigen Kapitel eingeführten Dateioperationen: Lesen, Schreiben (d.h. Anlegen, Ändern und Löschen) sowie Ausführen.

Eine typische Dateiinformation für eine Datei nach dem Befehl `ls -l` ist

`-rw-r-xrw-`

Der Erste Bindestrich wäre bei einem Verzeichnis ein `d` statt einem Bindestrich. Danch sind immer drei Buchstaben - jeweils `r` (*read*), `w` (*write*) und `x` (*execute*) (In Sonderfällen aus ein `s` (*superuser*)) - bzw. ein Bindestrich gelistet. Der letzte Bindestrich[14] kennzeichnet Sonderrechte.
Dabei bedeutet der Bindestrich *es ist kein Recht für diese Aktion erteilt* und der jeweilige Buchstabe *es ist ein Recht für diese Aktion erteilt*.

Die Aktionen haben dabei leicht unterschiedliche Auswirkungen, je nachdem ob es sich um eine Datei oder ein Verzeichnis handelt.

Bei Dateien gilt folgendes:

Aktion	**Beschreibung**
`r` - Lesen	Die Datei darf gelesen werden. (z.B. mit `cat` , `less` , etc.)
`w` - Ändern	Der Inhalt der Datei darf verändert werden .
`x` - Ausführen	Ist die Datei ein Skript oder eine Binärdatei, darf sie ausgefürt werden.

[14] bzw. auch hier wieder ein `s`

Bei Verzeichnissen gilt hingegen:

Aktion	**Beschreibung**
`r` - Lesen	Die beinhalteten Dateien und Ordner dürfen von angezeigt werden. (z.B. per `ls`)
`w` - Ändern	Innerhalb des Verzeichnisses dürfen Dateien angelegt, umbenannt und gelöscht werden.
`x` - Ausführen	Das Verzeichnis darf betreten werden

Die Sonderfälle `s` werden hier nicht besprochen, da sie für dem Umfang des ITS Praktikums für Sie nicht von Relevant sein sollten. Bei Interesse finden sich im Internet genügend Ressourcen.

Um die Eigentümer (Benutzer sowie Gruppe) eines Verzeichnisses bzw. einer Datei zu ändern, verwenden Sie den Befehl `chown` (*change owners*) und für die Änderung der Rechte den Befehl `chmod` (*change mode*). Die genaue Benutzung der Befehle finden Sie in den man pages.

11 Schlusswort

Wie Sie eventuell gemerkt haben, bin ich von einer anfänglich sehr ausführlichen Beschreibung jeder einzelnen Schritte und möglichen bzw. gängigen Optionen für Programme immer mehr zu einem banalen "*Sieh in der man page nach*" übergegangen.
Wenn Sie das Skript von Vorne bis Hinten durchgearbeitet haben, sollten Sie damit keine Probleme gehabt und sich so nach und nach in ein gängiges Szenario der Selbsthilfe eingearbeitet haben.
Wenn Sie das nächste mal einem unbekannten Programm gegenüberstehen, wissen Sie nun, wie Sie sich helfen können!

Hier noch ein paar letzte Tipps, wo Sie sich Hilfe beschaffen können:

- `man` , `whatis` , etc.
- google: offensichtlich...
- IRC: Manche denken eventuell, dass das IRC Netzwerk mittlerweile antik und eingestaubt ist. Dem kann ich nur widersprechen! Beispielsweise ist freenode.net ein nach wie vor sehr aktiv genutzter IRC-Server mit vielen programmier-spezifischen Channeln. Installieren Sie sich z.B. HexChat, betreten Sie channel wie #linux oder #bash und als oberste Regel beachten sie **immer**: Seien sie höflich und geduldig, sonst werden Sie auf diesem Wege scheitern.
- stackoverflow.com
- Sonstige Foren
- Fachliteratur: Bücher haben einen entschiedenen Vorteil gegenüber Ressourcen aus dem Internet: Während im Internet teilweise auch Halbwahrheiten (die der Unwissenheit der Autoren geschuldet sind) stehen, sind die Inhalte in Büchern in der Regel von Experten auf dem jeweiligen Gebiet geschrieben und mehrfach probe gelesen, um sicher zu stellen, dass alles stimmt. Die Bibliothek an der h_da hat sicherlich auch einiges an linuxbezogener Fachliteratur im Angebot und ist damit sogar kostenfrei für Sie zugänglich.