M|4.7: Linux-Kommandozeilenbefehle

Quelle: Cisco Netacad - Linux Unhatched - Stand 10.10.2023

Befehlssyntax

Die Befehlszeile stellt ein leistungsfähiges Werkzeug dar und ist häufig die vorrangige Methode zur Verwaltung einer Vielzahl von Systemen, angefangen bei kleinen, stromsparenden Geräten bis hin zu hochleistungsfähigen Cloud-Computing-Servern und allem dazwischen. Ein grundlegendes Verständnis der Befehlszeile ist unverzichtbar für die Diagnose und Reparatur der meisten Linux-basierten Systeme. Angesichts der weitreichenden Verbreitung von Linux können selbst diejenigen, die nicht hauptsächlich mit Linux-Systemen arbeiten, von einem grundlegenden Verständnis der Befehlszeile profitieren.

Was genau ist ein Befehl? Ein Befehl stellt ein Softwareprogramm dar, das, wenn es auf der Befehlszeilenschnittstelle (CLI) ausgeführt wird, eine bestimmte Aktion auf dem Computer ausführt. Durch die Eingabe eines Befehls wird ein Prozess vom Betriebssystem gestartet, der Eingaben verarbeitet, Daten manipuliert und Ausgaben generiert. Um einen Befehl auszuführen, geben Sie zunächst den Namen des Befehls ein.

Geben Sie bspw. `ls` (Kleinbuchstaben **L** und **S**) ein und drücken Sie die **Eingabetaste**, erhalten Sie folgende Ausgabe:

```
pi@herr-nm:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

Häufig leitet sich der Name eines Befehls davon ab, was er bewirkt oder wie der Entwickler die Funktion des Befehls am besten beschreiben möchte. Ein Beispiel hierfür ist der Befehl `ls`, der eine Liste von Informationen zu Dateien anzeigt. Indem Sie den Namen eines Befehls mit seiner Funktion verknüpfen, können Sie dabei unterstützt werden, sich die Befehle leichter zu merken.

!!! note “Hinweis”

```
Jeder Teil des Befehls wird normalerweise Groß- und Kleinschreibung beachtet, daher ist `LS` falsch und schlägt fehl, aber `ls` ist korrekt und wird ausgeführt.

pi@herr-nm:~$ LS
-bash: /usr/games/LS: Permission denied
```

Die meisten Befehle folgen einem einfachen Syntaxmuster:

```
Befehl [Optionen…] [Argumente…]
```

Befehle werden eingegeben, gefolgt von Optionen und/oder Argumenten, bevor man die Eingabetaste drückt. In der Regel beeinflussen Optionen das Verhalten des Befehls, während Argumente die Elemente oder Werte darstellen, auf die der Befehl angewendet

werden soll. Obwohl es in Linux einige Befehle gibt, die nicht vollständig dieser Syntax entsprechen, verwenden die meisten Befehle diese Struktur.

Im vorherigen Beispiel wurde der Befehl `ls` ohne Optionen oder Argumente ausgeführt. In diesem Fall resultiert das Standardverhalten darin, eine Liste der Dateien im aktuellen Verzeichnis zurückzugeben.

Argumente

Ein Argument dient dazu, dem Befehl Informationen zu übermitteln, auf den er angewendet werden soll. Beispielsweise kann der Befehl `ls` mit einem Verzeichnisnamen als Argument aufgerufen werden, um den Inhalt dieses Verzeichnisses anzuzeigen. Im folgenden Beispiel wird das Verzeichnis "Documents" als Argument verwendet:

```
pi@herr-nm:~$ ls Documents
MMBbS            ahorn.txt      esche.txt     linuxbefehle.txt
Privat           birke.txt      script.sh     zeder.txt
```

Die resultierende Ausgabe ist eine Liste der Dateien, die im Verzeichnis `Documents` enthalten sind.

Optionen

Optionen werden eingesetzt, um das Verhalten eines Befehls zu modifizieren. In der vorherigen Erklärung wurde der Befehl `ls` genutzt, um den Inhalt eines Verzeichnisses aufzulisten. Im folgenden Beispiel wird dem Befehl `ls` die Option `-l` hinzugefügt, was zu einer „long display"-Ausgabe führt. Das bedeutet, dass die Ausgabe mehr Informationen zu jeder aufgelisteten Datei bereitstellt.

```
pi@herr-nm:~$ ls -l
total 12
drwx------ 2 pi    pi   4096 Dec 20  2017 Desktop
drwx------ 4 pi    pi   4096 Dec 20  2017 Documents
drwx------ 2 pi    pi   4096 Dec 20  2017 Downloads
drwx------ 2 pi    pi   4096 Dec 20  2017 Music
drwx------ 2 pi    pi   4096 Dec 20  2017 Pictures
drwx------ 2 pi    pi   4096 Dec 20  2017 Public
drwx------ 2 pi    pi   4096 Dec 20  2017 Templates
drwx------ 2 pi    pi   4096 Dec 20  2017 Videos
```

```
Im Beispiel wurde `-l`, also ein kleines L verwendet.
```

Mehrere Optionen können gleichzeitig verwendet werden, entweder als separate Optionen wie in `-l -r` oder kombiniert wie `-lr` . Das `-r` steht im Bezug zu `ls` für *reverse* und gibt die umgedrehte alphabetische Reihenfolge aus. Die Ausgabe all dieser Beispiele wäre gleich:

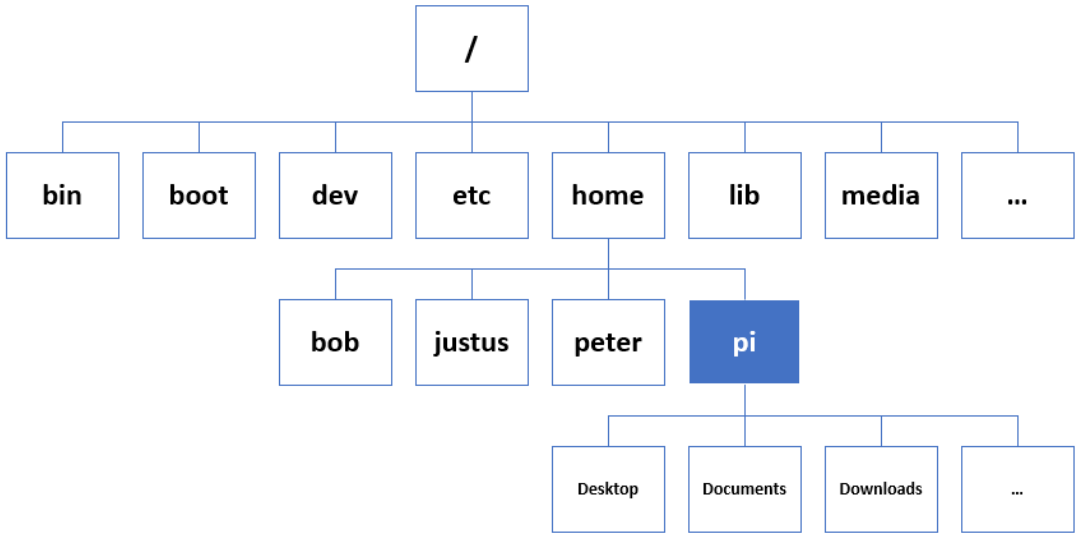
```
ls -l -r
ls -rl
ls -lr
```

Aktuelles Arbeitsverzeichnis

Um herauszufinden, wo Sie sich gerade im Dateisystem befinden, kann der Befehl `pwd` verwendet werden. Der Befehl `pwd` gibt das Arbeitsverzeichnis aus, Ihren aktuellen Aufenthaltsort im Dateisystem:

```
pi@herr-nm:~$ pwd
/home/pi
```

Das erste Verzeichnis, auch Wurzelverzeichnis wird root genannt und als `/` dargestellt. der im Beispiel angezeigte Ordner liegt also wie in der nachfolgenden Abbildung aufgeführt vor:



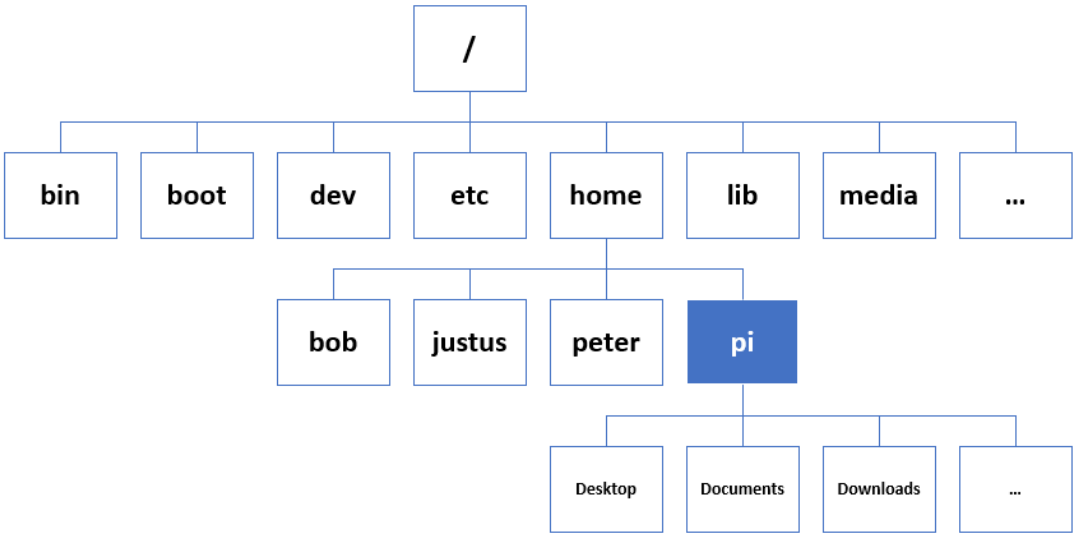

```
Im Ausgabe-Beispiel steht im Prompt eine Tilde `~`:

pi@herr-nm:~$ pwd
/home/pi

Diese steht für das Home-Verzeichnis des aktuellen Users `pi` also
`/home/pi`.
```

Verzeichnisse wechseln

Die folgende Abbildung wurde auch schon im vorherigen Abschnitt herangezogen:



Der Abbildung können Sie entnehmen, dass Sie sich im Verzeichnis `/home/pi` befinden. Darunter liegt das Verzeichnis `Downloads`. Um in das Verzeichnis `Downloads` zu wechseln, verwenden Sie es als Argument für den Befehl `cd`:

```
pi@herr-nm:~$ cd Downloads
pi@herr-nm:~/Downloads$
```

```
Wir sehen, dass der Prompt vor dem `$`-Zeichen nun den Pfad `~/Downloads` anzeigt. Dies ist das aktuelle Verzeichnis, in dem wir uns befinden. Es kann auch wie folgt beschrieben werden: `/home/pi/Downloads`.
```

Um in das Wurzelverzeichnis, also das oberste Verzeichnis im Linux-Dateisystem zu wechseln, benutzen Sie den Befehl `cd /`. Egal wo Sie sich im Dateisystem befinden, führen die Befehle `cd` (ohne Argument oder Option) oder auch `cd ~` in Ihr eigenes User-Home-Verzeichnis.

Pfade

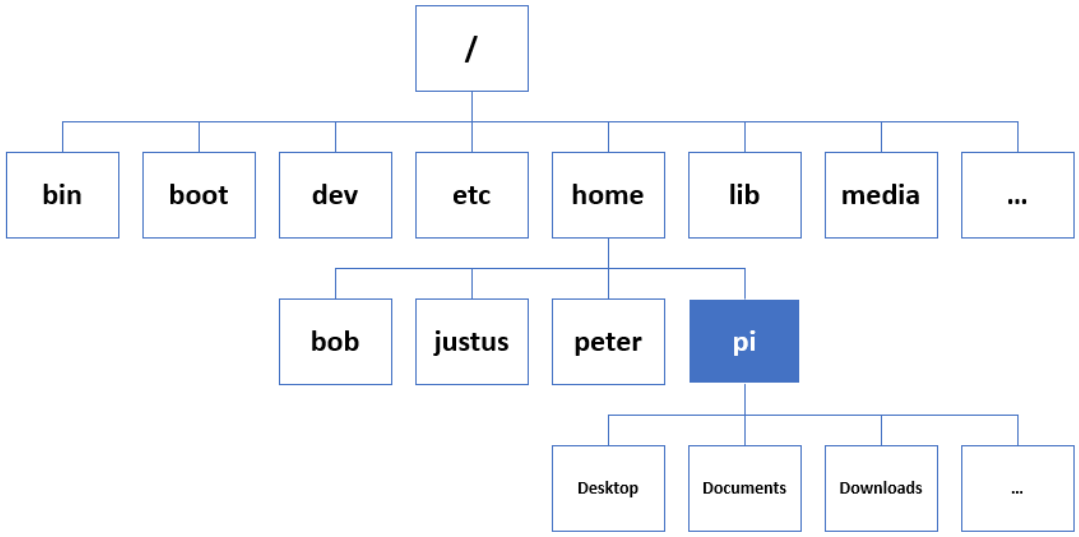
Absolute Pfade

Durch die Verwendung eines absoluten Pfades können Sie den exakten Speicherort eines Verzeichnisses angeben. Ein absoluter Pfad beginnt immer im Stammverzeichnis und wird daher stets mit dem Schrägstrich `/` eingeleitet. Ein Beispiel für einen absoluten Pfad ist der Pfad zum Home-Verzeichnis `/home/pi`. Dieser Pfad beginnt im Stammverzeichnis `/`, setzt sich fort ins `home`-Verzeichnis und schließlich ins Verzeichnis `pi`.

Ein absoluter Pfad wird immer in der Struktur des Dateisystems von oben nach unten adressiert. Dabei ist es irrelevant, in welchem Verzeichnis Sie sich aktuell befinden. Unter Windows wäre dies vergleichbar, wenn Sie den Speicherort `C:\Users\Bob\Desktop` benennen würden.

Relative Pfade

Ein relativer Pfad gibt den Weg zu einer Datei relativ zu Ihrem aktuellen Standort im Dateisystem an. Im Gegensatz zu absoluten Pfaden beginnen relative Pfade nicht mit dem Zeichen `/`. Stattdessen starten sie mit dem Namen eines Verzeichnisses. Werfen Sie erneut einen Blick auf das erste Beispiel des `cd`-Befehls. Das Argument stellt ein Beispiel für den einfachsten relativen Pfad dar: den Namen eines Verzeichnisses in Ihrem aktuellen Arbeitsverzeichnis.



Angenommen Sie befinden sich in einem Ordner `Linux`, der auf dem Desktop liegt. Das der absolute Pfad wäre dann `/home/pi/Desktop/Linux`. Nun angenommen, im Ordner `Downloads` liegt ein weiterer Ordner mit dem Namen `Games`. Wenn wir von dem Linux in den Games Ordner wechseln wollen, wird dies als relative Pfad-Angabe wie folgt aussehen:

```
pi@herr-nm:~$ cd ../../Downloads/Games
pi@herr-nm:~/Downloads/Games$
```

```
Der zweifache Punkt `..` steht für "ein Ordner höher" und in diesem Fall für
das Verzeichnis `/home/pi`. Schrittweise gelesen würde der Befehl wie folgt
abgearbeitet:

1. Wir befinden uns in `/home/pi/Desktop/Linux`.
2. Wir gehen einen Ordner nach oben in `/home/pi/Desktop/`. (erstes `..`)
3. Wir gehen einen Ordner nach oben in `/home/pi/`. (zweites `..`)
4. Wir gehen in den Ordner `Downloads`, der im Verzeichnis `/home/pi/` liegt.
5. Wir gehen in den Ordner `Games`, der in dem Verzeichnis `Downloads` liegt.
```

Dateien auflisten

Mit `ls` werden die Dateien und Ordner eines Verzeichnisses ausgegeben. Um Details zu einer Datei zu erfahren, wie zum Beispiel den Dateityp, die Berechtigungen, Eigentümer oder den Zeitstempel, führen Sie eine lange Auflistung durch, indem Sie die Option `-l` für den Befehl `ls` verwenden. Im Folgenden wird eine Auflistung des Verzeichnisses `/var/log` als Beispiel verwendet, da es eine Vielzahl von Ausgaben bereitstellt:

```
pi@herr-nm:~$ ls -l /var/log/
total 844
-rw-r--r-- 1 root   root  18047 Dec 20  2017 alternatives.log
drwxr-x--- 2 root   adm    4096 Dec 20  2017 apache2
drwxr-xr-x 1 root   root   4096 Dec 20  2017 apt
-rw-r----- 1 syslog adm    1346 Oct  2 22:17 auth.log
-rw-r--r-- 1 root   root  47816 Dec  7  2017 bootstrap.log
-rw-rw---- 1 root   utmp      0 Dec  7  2017 btmp
-rw-r----- 1 syslog adm     547 Oct  2 22:17 cron.log
-rw-r----- 1 root   adm   85083 Dec 20  2017 dmesg
-rw-r--r-- 1 root   root 325238 Dec 20  2017 dpkg.log
-rw-r--r-- 1 root   root  32064 Dec 20  2017 faillog
drwxr-xr-x 2 root   root   4096 Dec  7  2017 fsck
-rw-r----- 1 syslog adm     106 Oct  2 19:57 kern.log
-rw-rw-r-- 1 root   utmp 292584 Oct  2 19:57 lastlog
-rw-r----- 1 syslog adm   19573 Oct  2 22:57 syslog
drwxr-xr-x 2 root   root   4096 Apr 11  2014 upstart
-rw-rw-r-- 1 root   utmp    384 Oct  2 19:57 wtmp
```

Jede Zeile entspricht einer Datei, die im Verzeichnis enthalten ist. Die Informationen können in Felder unterteilt werden, die durch Leerzeichen getrennt sind. Die Felder sind wie folgt:

Dateityp

```
-rw-r--r-- 1 root   root  18047 Dec 20  2017 alternatives.log

drwxr-x--- 2 root   adm    4096 Dec 20  2017 apache2
```

Das erste Feld enthält tatsächlich zehn Zeichen, wobei das erste Zeichen den Dateityp angibt und die nächsten neun Berechtigungen angeben. Die Dateitypen sind:

Symbol	Dateityp	Beschreibung
`d`	Verzeichnis	Eine Datei die benutzt wird um darin andere Dateien zu speichern.
`-`	Reguläre Datei	Beinhaltet lesbare Dateien, Bilder, Binärdateien und komprimierte Dateien.
`l`	Symbolischer Link (Softlink)	Zeiger auf eine andere Datei.
`s`	Socket	Ermöglicht die Kommunikation zwischen Prozessen.
`p`	Pipe	Ermöglicht die Kommunikation zwischen Prozessen.

Symbol	Dateityp	Beschreibung
b	Blockbasiertes Gerät	Ermöglicht die Kommunikation mit Hardware.
c	Zeichenbasiertes Gerät	Ermöglicht die Kommunikation mit Hardware.

Die erste Datei `alternatives.log` ist eine reguläre Datei `-` , während die zweite Datei `apache2` ein Verzeichnis `d` ist.

Berechtigungen

```
drwxr-xr-x 2 root   root   4096 Apr 11  2014 upstart
```

Berechtigungen geben an, wie bestimmte Benutzer auf eine Datei zugreifen können. Lesen Sie weiter, um mehr über Berechtigungen zu erfahren.

Anzahl harter Verknüpfungen (Hardlinks)

```
-rw-r----- 1 syslog adm    1346 Oct  2 22:17 auth.log
```

Diese Zahl gibt an, wie viele Hardlinks auf diese Datei verweisen. Hardlinks sind nicht Thema dieses Moduls, werden aber im NDG Linux Essentials Kurs behandelt.

Benutzereigentümer

```
-rw-r----- 1 syslog adm     106 Oct  2 19:57 kern.log
```

Benutzer `syslog` besitzt diese Datei. Jedes Mal, wenn eine Datei erstellt wird, wird der Besitz automatisch dem Benutzer zugewiesen, der sie erstellt hat.

Gruppeneigentümer

```
-rw-rw-r-- 1 root   utmp 292584 Oct  2 19:57 lastlog
```

Gibt an, welche Gruppe diese Datei besitzt

Dateigröße

```
-rw-r----- 1 syslog adm   19573 Oct  2 22:57 syslog
```

Verzeichnisse und größere Dateien können in Kilobyte angezeigt werden, da die Anzeige ihrer Größe in Bytes eine sehr große Zahl darstellen würde. Daher kann es sich bei einem Verzeichnis um ein Vielfaches der Blockgröße handeln, die für das Dateisystem verwendet wird. Blockgröße ist die Größe einer Reihe von Daten, die im Dateisystem gespeichert sind.

Zeitstempel

```
drwxr-xr-x 2 root   root   4096 Dec  7  2017 fsck
```

Dies gibt den Zeitpunkt an, zu dem der Inhalt der Datei zuletzt geändert wurde.

Dateiname

```
-rw-r--r-- 1 root   root  47816 Dec  7  2017 bootstrap.log
```

Das letzte Feld enthält den Namen der Datei oder des Verzeichnisses.

Der Befehl sudo

Der Befehl `sudo` ermöglicht es einem Benutzer, einen Befehl als ein anderer Benutzer auszuführen, ohne eine neue Shell zu erstellen. Wenn Sie stattdessen einen Befehl mit Administratorrechten ausführen möchten, verwenden Sie ihn als Argument für den Befehl `sudo`. Wie der Befehl `su` geht der Befehl `sudo` standardmäßig davon aus, dass das Benutzerkonto `root` zum Ausführen von Befehlen verwendet werden soll.

Der Befehl `sudo` bietet nur administrativen Zugriff für eine einzelne Ausführung des angegebenen Befehls. Dies ist ein Vorteil, da es das Risiko reduziert, dass ein Benutzer versehentlich einen Befehl als `root` ausführt. Die Absicht, einen Befehl auszuführen, ist klar; der Befehl wird als `root` ausgeführt, wenn der Befehl `sudo` vorangestellt ist. Andernfalls wird der Befehl als normaler Benutzer ausgeführt.

Berechtigungen

Berechtigungen bestimmen, wie verschiedene Benutzer mit einer Datei oder einem Verzeichnis interagieren können. Wenn Sie eine Datei mit dem Befehl `ls -l` auflisten, enthält die Ausgabe Berechtigungsinformationen. Für das Beispiel werden wir ein Skript namens `hello.sh` im Verzeichnis `Documents` verwenden:

```
pi@herr-nm:~/Documents$ ls -l hello.sh
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Dateitypfeld

```
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Das erste Zeichen dieser Ausgabe gibt den Dateityp an. Erinnern Sie sich, wenn das erste Zeichen ein `-` ist, dann handelt es sich um eine normale Datei. Wenn das Zeichen ein `d` wäre, dann wäre es ein Verzeichnis.

Berechtigungsfeld

```
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Nach dem Dateitypzeichen werden die Berechtigungen angezeigt. Die Berechtigungen sind in drei Bereiche von je drei Zeichen unterteilt:

Eigentümer

```
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Der erste Bereich ist für den Benutzer, der die Datei besitzt. Wenn Ihr aktuelles Konto der Besitzer der Datei ist, wird der erste Bereich der drei Berechtigungen angewendet, und die anderen Berechtigungen haben keine Auswirkungen.

Der Benutzer, dem die Datei gehört und für den diese Berechtigungen gelten, kann durch das Feld “Eigentümer” bestimmt werden:

```
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Gruppe

```
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Der zweite Bereich ist für die Gruppe, die die Datei besitzt. Wenn Ihr aktuelles Konto nicht der Besitzer der Datei ist und Sie Mitglied der Gruppe sind, die die Datei besitzt, werden die Gruppenberechtigungen angewendet, und die anderen Berechtigungen haben keine Auswirkungen.

Die Gruppe für diese Datei kann durch das Feld “Eigentümer-Gruppe” bestimmt werden:

```
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Andere

```
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Der letzte Bereich ist für alle anderen Personen, für die diese ersten beiden Berechtigungsbereiche nicht gelten. Wenn Sie nicht der Benutzer sind, der die Datei besitzt, oder ein Mitglied der Gruppe, der die Datei besitzt, gilt der dritte Bereich von Berechtigungen für Sie.

Berechtigungstypen

Es gibt drei verschiedene Berechtigungen, die für eine Datei oder ein Verzeichnis platziert werden können: Lesen, Schreiben und Ausführen. Die Art und Weise, in der diese Berechtigungen gelten, unterscheidet sich für Dateien und Verzeichnisse, wie in der folgenden Tabelle dargestellt:

Berechtigung	Effekt auf eine Datei	Effekt auf ein Verzeichnis
lesen - read (`r`)	Erlaubt das Lesen oder das Kopieren von Datei-Inhalten.	In Zusammenhang mit dem Ausfühungsrecht wird ein detailliertes Listing bei ls -l angeboten. Ohne das Ausführungsrecht kann nur eine nicht detaillierte Auflistung erfolgen.
schreiben - write (`w`)	Erlaubt das Modifizieren oder Überschreiben von Dateien. Die Datei kann auch entfernt werden.	Damit diese Berechtigung funktioniert, muss zusätzlich das Ausführungsrecht gesetzt sein.
ausführen (`x`)	Ermöglicht die Ausführung einer Datei als Prozess. Hier ist zusätzlich auch eine Leseberechtigung erfordern.	Ermöglicht einem Benutzer, in das Verzeichnis zu wechseln sofern das Elternverzeichis ebenfalls eine Ausführungsberechtigung hat.

Dateiberechtigungen ändern

Der Befehl `chmod` (change mode) wird verwendet, um die Berechtigungen einer Datei oder eines Verzeichnisses zu ändern. Nur der `root`-Benutzer oder der Eigentümer einer Datei können deren Berechtigungen ändern.

Es gibt zwei Varianten, um Berechtigungen mit dem Befehl chmod zu ändern: symbolisch und oktal. Die symbolische Methode eignet sich gut, um jeweils einen Satz von Berechtigungen zu ändern. Die oktale oder numerische Methode erfordert Kenntnisse des oktalen Werts jeder der Berechtigungen und erfordert jedes Mal, dass alle drei Gruppen von Berechtigungen (Benutzer, Gruppe, andere) angegeben werden. Der Einfachheit halber wird nur die symbolische Methode erläutert.

Um die symbolische Variante von `chmod` zu verwenden, geben Sie zuerst an, welcher Satz von Berechtigungen geändert wird:

```
chmod [<SET><ACTION><PERMISSIONS>]... FILE
```

Symbol	Bedeutung
`u`	User: Der Eigentümer der Datei.
`g`	Group: Die Gruppe der die Datei gehört.
`o`	Others: Alle Anderen die nicht Eigentümer der Datei oder Mitglieder der Eigentümer-Gruppe sind.
`a`	All: Bezieht sich auf den Eigentümer, die Gruppe und alle Anderen gemeinsam.

Geben Sie als Nächstes ein Aktionssymbol an:

```
chmod [<SET><ACTION><PERMISSIONS>]... FILE
```

Symbol	Bedeutung
`+`	Setze die Berechtigung, sofern nicht vorhanden
`=`	Setzt genau die angegebene Berechtigung
`-`	Entfernt die Berechtigung, sofern vorhanden

Geben Sie nach einem Aktionssymbol eine oder mehrere Berechtigungen an, die bearbeitet werden sollen.

```
chmod [<SET><ACTION><PERMISSIONS>]... FILE
```

Symbol	Bedeutung
`r`	read - Leserecht
`w`	write - Schreibrecht
`x`	execute - Ausführungsrecht

Schließlich ein Leerzeichen und die Pfadnamen für die Dateien, um diese Berechtigungen zuzuweisen.

```
chmod [<SET><ACTION><PERMISSIONS>]... FILE
```

Die Datei `hello.sh`, die in den Beispielen auf der vorherigen Seite verwendet wird, ist ein Skript. Ein Skript ist eine Datei, die ausgeführt werden kann, ähnlich wie ein Befehl:

```
pi@herr-nm:~/Documents$ ls -l hello.sh
-rw-r--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Derzeit ist die Ausführungsberechtigung jedoch für keine der Berechtigungsgruppen festgelegt. Es ist kein `x` gesetzt.

Verwenden Sie den Befehl `chmod` mit dem Zeichen `u`, um die Berechtigungen des Eigentümers auszuwählen, das Zeichen `+`, um festzulegen, dass eine Berechtigung hinzugefügt wird, und das Zeichen `x`, um das Ausführungsrecht zu verwenden Damit sieht der Befehl wie folgt aus:

```
pi@herr-nm:~/Documents$ chmod u+x hello.sh
```

Keine Ausgabe zeigt an, dass der Befehl erfolgreich war. Kontrollieren Sie den Erfolg, indem Sie die Berechtigungen mit dem Befehl `ls -l` überprüfen:

```
pi@herr-nm:~/Documents$ ls -l hello.sh
-rwxr--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Der Eigentümer `root` hat nun das Ausführungsrecht und kann das Script mit dem Befehl `./hello.sh` ausführen.

Eigentümer einer Datei ändern

Anfangs ist der Eigentümer einer Datei der Benutzer, der sie erstellt. Der Befehl `chown` wird verwendet, um den Eigentümer von Dateien und Verzeichnissen zu ändern. Das Ändern des Eigentümers erfordert Administratorzugriff. Ein normaler Benutzer kann diesen Befehl nicht verwenden, um den Eigentümer einer Datei zu ändern. Er kann nicht einmal sein eigenen Dateien einem anderen Benutzer zu übertragen. Der Befehl `chown` erlaubt jedoch auch das Ändern der Eigentümer-Gruppe durch den Root-Nutzer sowie durch den Eigentümer der Datei.

```
pi@herr-nm:~/Documents$ ls -l hello.sh
-rwxr--r-- 1 root root 647 Dec 20  2017 hello.sh
```

Damit `pi` zum Eigentümer des Skripts `hello.sh` wird, verwenden Sie `pi` als erstes Argument und `hello.sh` als zweites Argument. Vergessen Sie nicht, den Befehl `sudo` zu verwenden, um die notwendigen Administratorrechte zu erhalten.

```
pi@herr-nm:~/Documents$ sudo chown pi hello.sh
[sudo] password for pi:
```

Überprüfen Sie, ob sich der Besitzer des Benutzers geändert hat, indem Sie den Befehl ls -l ausführen. Verwenden Sie den Dateinamen als Argument, um die Ausgabe zu begrenzen:

```
pi@herr-nm:~/Documents$ ls -l hello.sh
-rwxr--r-- 1 pi root 647 Dec 20  2017 hello.sh
```

Anzeigen von Dateien

Anzeigen von Dateien

Es stehen einige Linux-Befehle zur Verfügung, um den Inhalt von Dateien anzuzeigen. Der Befehl `cat`, dessen Name von engl. “concatenate” = “verketten” kommt, wird oft verwendet, um den Inhalt kleiner Dateien schnell anzuzeigen.

Der Befehl `cat` zeigt den gesamten Inhalt der Datei an, weshalb er hauptsächlich für kleinere Dateien empfohlen wird, bei denen die Ausgabe begrenzt ist und kein Scrollen erforderlich ist. Um den Inhalt einer Datei mit dem Befehl `cat` anzuzeigen, geben Sie einfach den Befehl ein und verwenden Sie den Namen der Datei, die Sie anzeigen möchten, als Argument:

```
cat [OPTIONEN] [DATEI]
```

Es gibt auch Pager für größere Dokumente (z.B. `more` und `less`) oder auch die Möglichkeit, nur die (standardmäßig) zehn ersten oder letzten Zeilen einer Datei anzeigen zu lassen (`head` und `tail`).

Dateien kopieren

Der Befehl `cp` wird verwendet, um Dateien zu kopieren. Ähnlich wie beim `mv`-Befehl (verschieben oder umbenennen) benötigt er mindestens zwei Argumente: eine Quelle und ein Ziel.

```
pi@herr-nm:~/Documents$ cp /home/pi/Desktop/test.txt .
```

```
Der einfach Punkt `.` steht für das aktuelle Verzeichnis. Im Beispiel wird
also die Datei `test.txt` aus dem Verzeichnis `/home/pi/Desktop` in das
aktuelle Verzeichnis kopiert. Das aktuelle Verzeichnis ist im Prompt vorne zu
sehen `~/Documents` also ausgeschrieben `/home/pi/Documents`. In diesen
Ordner wird die Datei `test.txt` hinein kopiert.
```

Berechtigungen können Auswirkungen auf Dateiverwaltungsbefehle haben, z.B. den Befehl `cp`. Um eine Datei zu kopieren, müssen Sie über die Ausführungsberechtigung verfügen, um auf das Verzeichnis zuzugreifen, in dem sich die Datei befindet, und über die Leseberechtigung für die zu kopierende Datei.

Es ist auch notwendig, Schreib- und Ausführungsberechtigungen für das Verzeichnis zu haben, in das die Datei kopiert wird. Normalerweise gibt es zwei Orte, an denen Sie immer Schreib- und Ausführungsberechtigungen für das Verzeichnis haben sollten: Ihr Home-Verzeichnis und das Verzeichnis `/tmp` .

Dateien verschieben

Der Befehl `mv` wird verwendet, um eine Datei von einem Speicherort im Dateisystem an einen anderen zu verschieben.

```
mv QUELLE ZIEL
```

Der Befehl `mv` erfordert wie `cp` mindestens zwei Argumente. Das erste Argument ist die Quelle, also der Pfad zur Datei, die verschoben werden soll. Das zweite Argument ist das Ziel, also der Pfad, zu dem die Datei verschoben wird. Die zu verschiebenden Dateien werden manchmal als Quelle bezeichnet, und der Ort, an dem die Dateien platziert werden sollen, wird als Ziel bezeichnet.

```
pi@herr-nm:~/Documents$ mv /home/pi/Desktop/test.txt .
```

Das Beispiel verschiebt die Datei `text.txt` in das aktuelle Verzeichnis `~/Documents`. Das Beispiel ist analog zu dem vorherigen Kopieren, nur dass nun die Datei nach dem Verschieben im ursrpünglichen Ordner nicht mehr zur Verfügung steht.

```
Der Befehl `mv` kann mehrere Dateien verschieben, solange das letzte Argument
für den Befehl das Zielverzeichnis ist.
```

Mittels `mv` können Dateien auch umbenannt werden:

```
pi@herr-nm:~/Documents$ mv /home/pi/Desktop/test.txt
/home/pi/Desktop/projekt.txt
```

Das Beispiel benennt die `text.txt` in `projekt.txt` um.

Dateien entfernen

Der Befehl `rm` (remove) wird verwendet, um Dateien und Verzeichnisse zu löschen. Es ist dabei wichtig zu beachten, dass gelöschte Dateien und Verzeichnisse nicht wie bei desktop-orientierten Betriebssystemen in einen Papierkorb gelangen. Wenn eine Datei mit dem Befehl `rm` gelöscht wird, ist sie fast immer dauerhaft verschwunden.

```
rm [OPTIONEN] DATEI
```

Ohne irgendwelche Optionen wird der Befehl `rm` normalerweise verwendet, um reguläre Dateien zu entfernen:

```
pi@herr-nm:~/Documents$ rm linux.txt
```

Der Befehl `rm` ignoriert Verzeichnisse, die entfernt werden sollen. Um ein Verzeichnis zu löschen, verwenden Sie eine rekursive Option, entweder die Optionen `-r` oder `-R`. Seien Sie vorsichtig, da diese Optionen „rekursiv“ sind. Der Befehl löscht rekursiv alle Dateien und alle Unterverzeichnisse:

```
pi@herr-nm:~/Documents$ rm -R Hausaufgaben
```

Das Beispiel führt dazu, dass der Ordner `Hausaufgaben` mit allen Unterordnern und Dateien gelöscht wird.

```
Der Befehl `rm` entfernt Dateien dauerhaft, es gibt standardmäßig keinen
Papierkorb, aus dem Dateien oder Ordner wiederhergestellt werden könnten.
```

Herunterfahren

Mit dem Befehl `shutdown` wird ein Linux-System ordnungsgemäß heruntergefahren. Es wird ein Zeitpunkt benötigt, welcher das Herunterfahren spezifiziert und der Befehl muss mit administrativen Rechten ausgeführt werden:

```
pi@herr-nm:~# sudo shutdown now
```

Dieses Zeitargument kann das Wort `now` sein, eine Tageszeit im Format `hh:mm` oder die Anzahl der Minuten, um den Vorgang um `+minuten` zu verzögern.

Der Befehl `shutdown` verfügt auch über ein optionales Nachrichtenargument, das eine Meldung angibt, die in den Terminals aller Benutzer angezeigt wird. Zum Beispiel:

```
pi@herr-nm:~# sudo shutdown +1 "Goodbye World!"
```