Linux Befehle PDF

Viele alltägliche Aufgaben lassen sich unter **UNIX- bzw. Linux- Betriebssystemen** mithilfe von verschiedenen Befehlen schnell und einfach erledigen - Der **sichere Umgang mit dem Terminal** ist daher eine besonders wichtige Fähigkeit für jeden Linux-/UNIX-Benutzer.

In diesem Beitrag werden Sie einen umfassenden Überblick zu den wichtigsten dieser Befehle für UNIX / Linux erhalten. Die hier vorgestellten Befehle werden dazu in die Bereiche **Grundlagen**, **Systemadministration** und **Netzwerke** unterteilt. Im ersten Abschnitt **Grundlagen** finden Sie Befehle, die bei der Nutzung von Linux- bzw. UNIX- Betriebssystemen unabdingbar sind. Im Bereich **Systemadministration** lernen Sie Befehle kennen, die Sie bei Ihrer täglichen Administrationsarbeit benötigen. Schließlich folgt der Abschnitt **Netzwerke**, welcher sich besonders an Fortgeschrittene Benutzer richtet, aber auch Einsteigern einige hilfreiche Informationen übermittelt.

Inhaltsverzeichnis

Grundlagen

Ein Terminal öffnen

Bevor Sie mit der Verwendung der Befehle beginnen können, müssen Sie zunächst ein Terminal auf Ihrem Linux-/UNIX-System öffnen. Meist ist eine einfach Suche nach "terminal" über die Suchfunktion des Menü dabei ausreichend.

Auf manchen Linux-Distributionen haben Sie zusätzlich jedoch auch die Möglichkeit, über die Tastenkombination [Alt+F2] ein "Run"-Eingabefeld zu öffnen. Dort können Sie durch Eingabe von "gnome-terminal" beispielsweise das gnome-terminal öffnen.

```
gnome-terminal
```

An- und Abmeldung

login - Benutzer anmelden

Der Befehl login wird verwendet, um einen Benutzer anzumelden. Falls bereits ein anderer Benutzer angemeldet ist, wird dieser automatisch abgemeldet. Der Befehl wird normalerweise automatisch als Antwort auf die login:-Eingabeaufforderung im Terminal ausgeführt. Es folgt die Syntax des Befehls:

```
login [Optionen] [Benutzer]
```

Nach Ausführung des Befehls erfolgt eine Passwortabfrage für den Benutzer.

logout - Aktiven Benutzer abmelden

Der zurzeit angemeldete Benutzer kann mit dem Befehl logout wieder abgemeldet werden:

```
logout
```

whoami - Aktiven Benutzer anzeigen

Der Abruf des aktuellen Benutzers ist vor allem bei häufigem einem Wechsel von Identitäten sehr hilfreich. Der mit der aktiven Benutzer-ID (UID) verknüpfte Benutzername lässt sich dazu mit dem Befehl whoami ausgeben:

```
whoami
```

su - Benutzer wechseln

Der Befehl su ("switch user") wird in der Praxis immer wieder benötigt: Er ermöglicht es Ihnen, die Identität eines anderen Benutzers anzunehmen. Die Syntax des Befehls lautet grundsätzlich folgendermaßen:

```
su [Benutzername]
```

Bei Weglassen des Benutzernamen wird automatisch der root-Benutzer verwendet. Bevor Sie mit der Identität des jeweiligen Benutzers fortfahren können, erfolgt zudem eine Passwortabfrage.

Der Befehl wird vor allem bei Administrativen Aufgaben, die besondere root-Rechte benötigen immer wieder benötigt, um einen Wechsel auf den root-Benutzer zu realisieren.

sudo - Befehlsausführung im Namen anderer Benutzer

Mit dem Befehl sudo (su "do") können Befehle im Namen bzw. mit den Berechtigungen anderer Benutzer ausgeführt werden. Anwendung findet der Befehl somit oft dann, wenn bestimmte Aktionen ausgeführt werden sollen, die root-Rechte erfordern.

Damit ein Benutzer / eine Gruppe den Befehl sudo verwenden darf, muss er / sie in der Datei /etc/sudoers registriert werden. Standardmäßig wird hier die Gruppe "sudo" bereitgestellt, dessen Mitglieder alle den sudo-Befehl verwenden dürfen.

Die allgemeine Syntax von sudo lautet:

```
sudo [Optionen] [Befehl]
```

sudo kann konkret mit der Option -i auch verwendet werden, um eine Rootshell zu starten:

```
sudo -i
```

Nähere Informationen zu sudo, der /etc/sudoers und verfügbaren Optionen sowie auch Beispiele finden Sie in unserem Beitrag.

exit - Beenden von Sitzungen

Sitzungen, wie zum Beispiel Root-Sitzungen, die zuvor per su-Befehl gestartet wurden, oder auch SSH-Sitzungen, können mit dem Befehl exit wieder beendet werden. Genauso kann exit auch verwendet werden, um den aktiven Benutzer abzumelden - ähnlich wie

beim dem Befehl logout.

```
exit
```

Verzeichnisnavigation

Der Navigation von Verzeichnissen dient an erster Stelle der Befehl cd, welcher das Wechseln des aktuellen Arbeitsverzeichnisses ermöglicht. cd ist ein eingebauter Befehl der Shell und muss damit nicht zusätzlich installiert werden.

cd wird folgendermaßen angewendet:

```
cd [Optionen] [Verzeichnis]
```

Optionen werden für die meisten Anwendungen des Befehls cd nicht benötigt. Die Angabe eines Verzeichnisses erfolgt mithilfe absoluter oder relativer Verzeichnispfade. Wenn Sie mehr über die verschiedenen Möglichkeiten zur Anwendung des Befehls cd erfahren möchten, sehen Sie sich auch unseren dedizierten Beitrag dazu an.

Bei der Verzeichnisnavigation ist es auch wichtig, die Inhalte eines Verzeichnisses, etwa des aktuellen Arbeitsverzeichnisses, anzeigen zu können. Hierzu wird der Befehl ls verwendet:

```
ls [Optionen] [Datei / Verzeichnis]
```

Werden weder eine Datei, noch ein Verzeichnis angegeben, so gibt ls standardmäßig den Inhalt des aktuellen Arbeitsverzeichnisses, unter Auflistung der Datei- und / oder Verzeichnisnamen aus. Mithilfe absoluter oder relativer Verzeichnis- / Dateipfade können die Inhalte bestimmter anderer Verzeichnisse oder einzelne Dateien ausgegeben werden.

Eine der wichtigsten Optionen des Befehls ls ist die Option -l, welche die Ausgabe detaillierterer Informationen zu Berechtigungen, Zeitstempeln, Kapazitäten etc. ermöglicht:

```
ls -l [Datei / Verzeichnis]
```

Versteckte Dateien können mithilfe der Option -a angezeigt werden:

```
ls -a [Datei / Verzeichnis]
```

Nähere Informationen zum Befehl ls finden Sie in unserem Beitrag zu diesem Befehl.

Datei-/Verzeichnismanipulation

mkdir - Erstellen von Verzeichnissen

Mithilfe von mkdir ("make directory") können Sie neue Verzeichnisse erstellen, unter denen weitere Dateien und Verzeichnisse angelegt werden können:

```
mkdir [Optionen] [Pfad]
```

Wenn Sie einen ganzen Pfad von noch nicht existierenden, verschachtelten Verzeichnissen anlegen wollen, dann müssen Sie die Option -p zusätzlich übergeben. Fehlende Verzeichnisse des angegebenen Pfades werden bei der Ausführung automatisch angelegt.

```
mkdir -p [Pfad]
```

touch - Anlegen von Dateien / Aktualisieren der Zugriffszeit

Neue Dateien mit aktuellen Daten, wie beispielsweise der Zugriffszeit, können mit dem Befehl touch angelegt werden. touch legt jedoch nur dann neue Dateien der Länge null an, wenn die angegebene Datei noch nicht im jeweiligen Verzeichnis existiert. In solchen Fällen wird lediglich das Zugriffsdatum bzw. die Zugriffszeit der existierenden Datei aktualisiert. Die Syntax des Befehls lautet:

```
touch [Optionen] [Dateipfad]
```

rmdir - Löschen von Verzeichnissen

Der Befehl rmdir wird verwendet um Verzeichnisse zu löschen. Dabei ist zu beachten, dass nur leere Verzeichnisse gelöscht werden können.

```
rmdir [Optionen] [Pfad]
```

rm - Löschen von Dateien/Verzeichnissen

Einträge einer oder mehrerer Dateien können mit dem Befehl rm aus einem Verzeichnis gelöscht werden.

```
rm [Optionen] [Pfad]
```

Der Befehl kann außerdem auch verwendet werden, um rekursiv den gesamten Inhalt eines Verzeichnisses zu löschen. Dazu müssen Sie die Option -r verwenden:

```
rm -r [Pfad]
```

mv - Verschieben und Umbenennen

Je nach Anwendung lassen sich mit mv ("move") Dateien und Verzeichnisse verschieben und umbenennen. Verschieben lassen sich Dateien / Verzeichnisse durch Angabe des alten und neuen Pfades:

```
mv [alter Pfad] [neuer Pfad]
```

Beim Umbenennen einer Datei / eines Verzeichnisses bleiben der alte und der neue Pfad gleich. Lediglich der Datei-/Verzeichnisname wird verändert.

cp - Kopieren

Mit cp können Sie eine Datei in eine Datei mit einem anderen Namen, oder in ein anderes Verzeichnis kopieren. Außerdem haben Sie die Möglichkeit mehrere Dateien in ein bestimmtes Verzeichnis zu kopieren. Die Syntax des Befehls lautet:

```
cp [Quelle] [Ziel]
```

nano - Texteditor

Der Befehl nano ruft einen einfache Texteditor auf. Zum einen haben Sie die Möglichkeit, eine bestimmte existierende Datei durch Angabe des Dateipfads mit dem Texteditor zu bearbeiten. Zum anderen können Sie auch eine neue Datei erstellen und diese direkt bearbeiten - Dazu wird entweder kein Pfad angegeben oder die angegebene Datei existiert nicht.

```
nano [Dateipfad]
```

cat - Ausgeben von Dateiinhalten

Mit cat können Sie den Inhalt einer Datei direkt im Terminal ausgeben, ohne diese zu manipulieren. Der Pfad der Datei wird dazu einfach hinter dem Befehl angegeben:

```
cat [Dateipfad]
```

Da angegebene die Datei bei der Ausgabe durch cat nicht modifiziert werden kann, eignet sich der Befehl hervorragend dazu, den Inhalt einer Datei nach der Bearbeitung noch einmal zu überprüfen.

Suchen im Dateisystem

Eine Suche nach Dateien oder Verzeichnissen lässt sich unter Linux- bzw. UNIX- Systemen leicht mit dem Befehl find durchführen. Sie können dabei von beliebigen Startpunkten aus ganze Verzeichniszweige durchsuchen und die Ergebnisse zudem auch filtern.

Der Befehl stellt neben den einfachen Such- und Filterfunktionen auch noch einige speziellere Funktionen bereit, die Ihn zu einem mächtigen Universalwerkzeug machen. Wenn Sie mehr über diese erfahren wollen, lesen Sie auch unseren separaten Beitrag zum diesem Befehl.

find - Datei-/Verzeichnissuche

find kann mit einer Vielzahl Optionen ausgeführt werden, um verschiedene Daten für die Suche heranzuziehen und die Ergebnisse zusätzlich zu filtern. Die Syntax des Befehls lautet jedoch allgemein:

```
find [Startpunkt] [Optionen / Ausdruck]
```

Filtern nach Name

Bei einer einfachen Datei- oder Verzeichnissuche nach einem bestimmten Namen wird die Option -name verwendet. Sie benötigen außerdem den Startpunkt für die Suche, sowie die Suchbedingungen bzw. den exakten Namen der gesuchten Datei / des gesuchten Verzeichnisses:

```
find [Startpunkt] -name [Datei-/Verzeichnisname]
```

Wenn Sie die Groß- und Kleinschreibung bei der Suche ignorieren wollen, dann können Sie auch die Option -iname verwenden:

```
find [Startpunkt] -iname [Datei-/Verzeichnisname]
```

Bei der Angabe von Datei- bzw. Verzeichnisnamen ist es auch möglich, sogenannte Wildcards zu verwenden, um eine dynamische Suche nach mehreren Dateien durchzuführen. Eine genaue Kenntnis des Dateinamen ist somit auch nicht nötig. Wie Sie Wildcards in Ihrer Suche verwenden, erfahren Sie ebenfalls in unserem separaten Beitrag zum Befehl find.

Filtern nach Kapazität

Die Option -size wird verwendet, um Dateien und Verzeichnisse aufgrund ihrer Kapazität zu filtern. Die Syntax lautet hier:

```
find [Startpunkt] -size [+-][Kapazität][ckMG]
```

Bei der einfachen Angabe einer Kapazität werden alle Dateien/Verzeichnisse gesucht, die dieser genau entsprechen. Die Einheit können Sie dabei durch Anhängen der Zeichen 'c' für Byte, 'k' für Kilobyte, 'M' für Megabyte und 'G' für Gigabyte festlegen. Wenn Sie keine Einheit angeben, dann bezieht sich die Kapazität auf die Anzahl belegter Blöcke im Dateisysten.

Statt einer genauen Kapazitäts-Angabe können Sie schließlich auch ein Plus '+' oder Minus '-' voranstellen. Das '+' steht dann für "größer als", während das '-' für "kleiner als" steht.

Filtern nach Zugriffsrechten

Bei der Suche können Sie auch nach Dateien mit bestimmten Zugriffsrechten filtern. Dazu verwenden Sie die Option -perm und geben die Zugriffsrechte als Oktalzahl an:

```
find [Startpunkt] -perm [Zugriffsrechte als Oktalzahl]
```

Wie die Schreibweise als Oktalzahl genau funktioniert, können Sie in unserem Beitrag zum Linux-Berechtigungssystem nachlesen.

Filtern nach Besitzer

Mit der Option -user können Sie auch nach Dateien/Verzeichnissen mit einem bestimmten Besitzer suchen:

```
find [Startpunkt] -user [Benutzername]
```

Filtern nach Typ

Zuletzt können Sie mit -type auch noch nach Elementen mit bestimmter Typisierung suchen. Die wichtigsten Parameter für die Nutzung mit dieser Option sind 'f' für Dateien (file), 'd' für Verzeichnisse (directories) und 'l' für symbolische Links (link).

Systembefehle

shutdown - Herunterfahren / Neustart

Mit dem Befehl **shutdown** können Neustarts, und das Herunterfahren des Systems eingeleitet und durch bestimmte Optionen sogar zeitlich gesteuert werden.

Herunterfahren lässt sich das System mit der **Option -h**. Sie müssen zusätzlich außerdem einen Zeitpunkt festlegen, zu dem der Vorgang gestartet werden soll. Dieser kann entweder in Minuten nach Befehlsausführung oder im 24 Stunden Format **"hh:mm"** angegeben werden:

```
shutdown -h [Zeit]
```

Neustarts werden mit der **Option -r** (reboot) durchgeführt:

```
shutdown -r [Zeit]
```

Zuletzt gibt es noch die Möglichkeit, ein geplantes Herunterfahren / einen Neustart vorzeitig abzubrechen. Dazu verwenden Sie die **Option -c** (cancel):

```
shutdown -c
```

Unter Debian 10 Buster kann **shutdown**, aufgrund des vollständigen Wechsels auf **systemd**, nicht mehr ohne Weiteres eingesetzt werden. Sie können stattdessen auf den Befehl **systemctl** zurückgreifen, welcher im Folgenden einmal erklärt wird.

systemctl - Kontrolle des systemd System- und Service-Manager

Herunterfahren / Neustarten

Mit **systemctl** können Befehle an **systemd** gesendet werden, um beispielweise das System herunterzufahren oder neuzustarten. Ein Herunterfahren wird mit **poweroff** eingeleitet:

```
systemctl poweroff
```

Neustarts können Sie mit **reboot** durchführen:

```
systemctl reboot
```

Steuerung von Services / Units

Neben **poweroff** und **reboot** bietet das Befehlszeilenwerkzeug **systemctl** auch noch Funktionen für die Steuerung von Systemprogrammen, sogenannten Units.

Zunächst gibt es die beiden Funktionen **start** und **stop**, welche zum Starten und Stoppen der Units verwendet werden können:

```
systemctl [start / stop] [Unit]
```

Die Funktion **restart** wird verwendet, um Units neuzustarten. Dies ist hilfreich, wenn nach einer Änderung beispielsweise die Konfigurationsdateien eines Programms neu eingelesen werden sollen:

```
systemctl [restart] [Unit]
```

Schließlich gibt es noch die Funktion **status**, welche, wie der Name schon vermuten lässt, den Status einer Unit im Terminal ausgibt.

```
systemctl [status] [Unit]
```

Benutzer-/Gruppenverwaltung

In diesem Abschnitt werden Sie einige wichtige Befehle für die Verwaltung von Benutzern und Gruppen unter Linux- bzw. UNIX-Betriebssystemen kennenlernen. Wenn Sie Näheres zur Funktionsweise des Benutzer-/Gruppensystems und den Befehlen erfahren wollen, lesen Sie auch unseren ausführlichen Beitrag zu diesem Thema. Sie finden dort außerdem Beispielanwendungen für die Befehle.

Die meisten der folgenden Befehle erfordern für ihre Ausführung die Rechte des root-Benutzers.

Benutzer

adduser - Hinzufügen neuer Benutzer

Mit dem Befehl adduser können Sie neue Benutzer erstellen. Der Befehl benötigt dazu lediglich den Namen des zu erstellenden Benutzers, alle anderen notwendigen Daten werden automatisch abgefragt. Dazu gehört das Passwort, der vollständige Name des Benutzers, sowie einige optionale Daten wie die Zimmernummer, Telefon und Sonstiges.

```
adduser [Benutzername]
```

Neben dem Benutzer wird auch eine gleichnamige Gruppe und ein entsprechendes Home-Verzeichnis erstellt.

passwd - Ändern von Passwörtern

Das Ändern von Passwörtern ist mit dem Befehl passwd möglich: Der Root-Benutzer hat dabei die Möglichkeit, das Passwort eines jeden Benutzers zu ändern/aktualisieren. Der jeweilige Benutzername wird dazu einfach als Parameter übergeben. Jeder Benutzer kann außerdem sein eigenes Passwort ändern - dazu ist keine Angabe eines Benutzernamen notwendig.

```
passwd [Benutzername]
```

usermod - Bearbeiten von Benutzerkonten

Der Befehl usermod ermöglicht die Bearbeitung von Informationen und Gruppenzugehörigkeiten bereits existierender Benutzerkonten. Dabei kommen verschiedene Optionen zum Einsatz. Zunächst folgt jedoch die allgemeine Syntax des Befehls:

```
usermod [Optionen] [Benutzer]
```

Mit der Option -l können Sie einen bestehenden Benutzer umbenennen:

```
usermod -l [Neuer Benutzername] [Benutzer]
```

Benutzer können mit der Option -aG in Gruppen hinzugefügt werden:

```
usermod -aG [Gruppe/-n] [Benutzer]
```

deluser - Löschen von Benutzern

Wollen Sie einen Benutzer wieder löschen, dann kommt der Befehl deluser zum Einsatz. Die Syntax lautet:

```
deluser [Optionen] [Benutzer]
```

Wenn Sie dem Befehl keine besonderen Optionen übergeben, dann wird zwar der Benutzer gelöscht, das Home-Verzeichnis, sowie alle vom Benutzer erstellten Daten bleiben jedoch erhalten. Wenn Sie das Home-Verzeichnis ebenfalls löschen wollen, dann verwenden Sie daher die Option --remove-home:

```
usermod --remove-home [Benutzer]
```

Sie können mit der Option --remove-all-files zudem auch alle Dateien des Benutzers löschen:

```
usermod --remove-all-files [Benutzer]
```

Gruppen

addgroup - Hinzufügen neuer Gruppen

Neue Gruppen können Sie mit addgroup erstellen. Sie müssen dabei lediglich den Namen der Gruppe angeben:

```
addgroup [Gruppenname]
```

groupmod - Bearbeiten von Gruppen

Mit dem Tool groupmod können Sie bestehende Gruppen bearbeiten. Die Syntax ist dabei ähnlich wie bei usermod:

```
groupmod [Optionen] [Gruppe]
```

Sie haben auch hier wieder die Möglichkeit, verschiedene Optionen einzusetzen. Eine der wichtigsten Optionen des Befehls ist --new-name, welche für das Umbenennen von Gruppen verwendet wird:

```
groupmod --new-name [Gruppe]
```

delgroup - Löschen von Gruppen

Schließlich können Sie Gruppen mit delgroup auch wieder löschen:

```
delgroup [Optionen] [Gruppe]
```

Während Gruppen bei Ausführung ohne zusätzliche Optionen ohne jegliche Rückfrage gelöscht werden, gibt es hier noch die Option --only-if-empty, welche sicherstellt, dass Gruppen nur dann gelöscht werden, wenn sie auch wirklich leer sind - also keine Mitglieder mehr enthalten:

```
delgroup --only-if-empty [Gruppe]
```

Berechtigungen

Unter Linux- bzw- UNIX-Betriebssystemen bekommen sowohl Dateien, als auch Verzeichnisse alle einen Besitzer (user), sowie bestimmte Zugriffsrechte zugeordnet. Grundlegende Kenntnisse zu den verfügbaren Befehlen sind daher äußerst wichtig.

Sie lernen in diesem Abschnitt die beiden Befehle chmod und chown kennen. Ein detaillierteres Bild zur Funktionsweise des Berechtigungssystems erhalten Sie in unserem Beitrag "Das Linux-Berechtigungssystem".

chmod - Anpassen von Zugriffsrechten

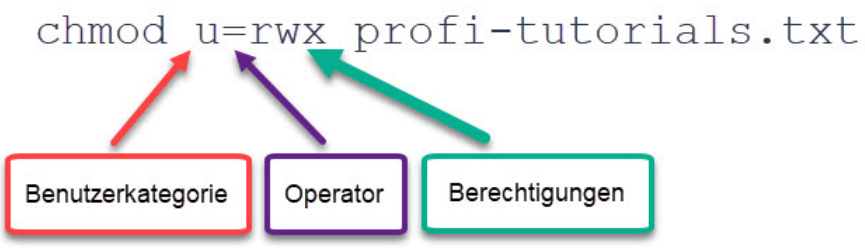
Der Befehl chmod (change mode) wird verwendet, um die Zugriffsrechte von Dateien anzupassen. Allgemein lautet die Syntax des Befehls folgendermaßen:

```
chmod [Optionen] [Berechtigungen / Modus] [Pfad]
```

Grundsätzlich gibt es 2 Methoden der Rechtevergabe mit dem Befehl: Die Symbolische- und die Numerische Rechtevergabe, welche nun einmal vorgestellt werden.

Symbolische Rechtevergabe

Bei der symbolischen Vorgehensweise werden die Zugriffsrechte auf Dateien oder Verzeichnisse mit den Zeichen 'r', 'w' und 'x' zugeordnet. 'r' steht dabei für lesenden Zugriff (read), 'w' für die Schreibrechte (write) und 'x' schließlich für das Ausführungsrecht (execute). Die Syntax für die symbolische Rechtevergabe sieht folgendermaßen aus:



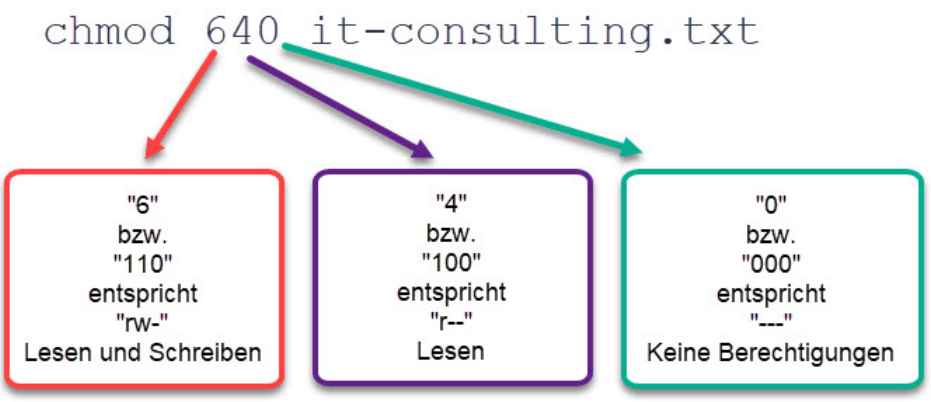
Zunächst wird die Benutzerkategorie ausgewählt, auf die die neuen Rechte angewendet werden sollen. Zugelassen sind die Kategorien 'u' für Benutzer (user), 'g' für Gruppe (group), 'o' für Andere (other) und zuletzt 'a' für alle (all). Anschließend folgt der Operator '-' um Rechte zu entfernen, '+' um Rechte zu erteilen und '=' um Rechte explizit zu setzen. Rechts neben dem Operator stehen dann schließlich die zu erteilenden Rechte in der Form "rwx".

Wenn Sie mehrere Änderungen an einer Datei / einem Verzeichnis mit einem Befehl vornehmen wollen, dann können Sie diese auch, durch Kommata getrennt, hintereinander auflisten.

Numerische Rechtevergabe

Bei der numerischen Vorgehensweise werden die Zugriffsrechte in Form einer dreistelligen Oktalzahl angegeben. Jede der drei Ziffern steht dabei für die Rechte einer bestimmten Benutzerkategorie: Die erste Ziffer stellt die Berechtigungen für den Besitzer dar,

die zweite für die Gruppe und die dritte für Andere.



Um die Berechtigungen abzulesen, die durch eine Ziffer dargestellt werden, muss diese in das Dualsystem konvertiert werden. Die resultierende dreistellige Dualzahl stellt die Berechtigungen dar.

Die erste Ziffer der Dualzahl steht dabei für den lesenden Zugriff, die zweite für das Schreibrecht und die dritte zuletzt für das Ausführungsrecht. Die 1 bedeutet dabei, dass das jeweilige Recht erteilt wurde, während eine 0 bedeutet, dass das Recht nicht erteilt wurde.

chown - Anpassen von Besitzer und Gruppe

Mit dem Befehl chown können Sie den Besitzer bzw. die Gruppe einer Datei oder eines Verzeichnisses festlegen. Die Syntax lautet allgemein:

```
chown [Optionen] [Besitzer]:[Gruppe] [Pfad]
```

Sie können den Besitzer bzw. die Gruppe auch einzeln zuweisen. Um nur den Benutzer zuzuweisen, lassen Sie den Teil ":[Gruppe]" einfach aus:

```
chown [Optionen] [Besitzer] [Pfad]
```

Wenn Sie nur eine Zuweisung der Gruppe benötigen, dann können Sie den Teil "[Besitzer]" auslassen. Wichtig dabei ist, dass der Doppelpunkt ':' stehen bleibt, da dieser signalisiert, dass es sich um den Gruppennamen, nicht jedoch um den Namen des Besitzers handelt.

History

Die auf einem Linux- / UNIX-System mit einem Benutzer ausgeführten Befehle werden in der sogenannten History protokolliert. Mit dem Befehl history ist es möglich, diese Aufzeichnungen auszulesen und Befehle erneut auszuführen. So ist es etwa möglich in Erfahrung zu bringen, welche Befehle ein Benutzer wann ausgeführt hat.

Die History ausgeben

Die History eines Benutzers kann durch die einfache Eingabe des Befehls history mit diesem Benutzer - ohne Optionen oder Parameter - ausgegeben werden:

```
history
```

Sollen nur die letzten n Eintragungen der history ausgegeben werden, so kann dem Befehl optional eine entsprechende Ganzzahl n angefügt werden:

```
history [Anzahl]
```

Befehle ausführen

Links neben jeder Eintragung der History finden Sie ihre jeweilige Nummer. Möchten Sie einen Befehl der History erneut ausführen, geben Sie diese Nummer sowie ein vorangestelltes Ausrufungszeichen "!" im Terminal ein. Der Befehl mit der entsprechenden Nummer wird dann ausgeführt, ohne dass eine komplette Neueingabe nötig ist:

```
![Nummer]
```

Achten Sie bei der Wiederausführung von Befehlen darauf, dass bei relativen Pfadangaben Ihr aktuelles Arbeitsverzeichnis als Startpunkt verwendet wird - nicht das Verzeichnis, in welchem Sie sich bei der ersten Ausführung befanden.

History löschen

Einzelne Befehle können mithilfe der Option -d und durch Angabe ihrer entsprechenden Nummer aus der History gelöscht werden:

```
history -d [Nummer]
```

Die komplette History ist dagegen mit der Option -c zu löschen:

```
history -c
```

Systemadministration

Prozessmanagement

In diesem Abschnitt "Prozessmanagement" lernen Sie einige der wichtigsten Befehle zur Erkennung und Behebung von Fehlern und System-Instabilitäten. Ist Linux bzw. UNIX zwar ein äußerst stabiles System, so kann es dennoch immer auch zu Fehlern kommen - die hier gezeigten Befehle helfen Systemadministratoren bei der Bewältigung von alltäglichen Problemen.

top - Prozessübersicht

Mit dem Befehl **top** können Sie sich eine dynamische Übersicht aller laufenden Prozesse, sowie die Auslastung diverser Systemressourcen anzeigen lassen. Durch Eingabe des folgenden Befehls gelangen Sie zur Übersicht:

```
top
```

Die erscheinende Übersicht sieht etwa folgendermaßen aus:

```
top - 15:34:07 up  3:45,  1 user,  load average: 0,00, 0,00, 0,00
Tasks: 156 total,   1 running, 155 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,7 us,  1,8 sy,  0,0 ni, 96,5 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  2033688 total,   736616 free,   697316 used,   599756 buff/cache
KiB Swap:  1952764 total,  1952764 free,        0 used.  1173020 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1592 hellberg  20   0 2264948 269280  83948 S   3,6 13,2   0:27.13 gnome-shell
 1498 hellberg  20   0  358856  52656  30104 S   2,3  2,6   0:04.48 Xorg
 1911 hellberg  20   0  601392  33264  24796 S   2,0  1,6   0:01.62 gnome-terminal-
  775 root      20   0  183124  13124  10556 S   0,7  0,6   0:19.60 vmtoolsd
 2356 root      20   0   44912   3644   3020 R   0,7  0,2   0:00.05 top
 1744 hellberg  20   0  252388  28104  24904 S   0,3  1,4   0:20.42 vmtoolsd
    1 root      20   0  139052   6932   5268 S   0,0  0,3   0:02.16 systemd
    2 root      20   0       0      0      0 S   0,0  0,0   0:00.02 kthreadd
    3 root      20   0       0      0      0 S   0,0  0,0   0:00.23 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0,0  0,0   0:00.68 rcu_sched
    8 root      20   0       0      0      0 S   0,0  0,0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0,0  0,0   0:00.00 migration/0
   10 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S   0,0  0,0   0:00.04 watchdog/0
   12 root      20   0       0      0      0 S   0,0  0,0   0:00.00 cpuhp/0
   13 root      20   0       0      0      0 S   0,0  0,0   0:00.00 cpuhp/1
   14 root      rt   0       0      0      0 S   0,0  0,0   0:00.04 watchdog/1
```

Genauere Informationen zu diesem Befehl finden Sie auch hier.

ps - Weitere Prozessübersicht

Eine Alternative für die Anzeige von Prozessen ist der Befehl **ps** (processes). Bei Eingabe des Befehls ohne Angabe zusätzlicher Parameter werden dabei die momentan im Terminal geöffneten Prozesse aufgelistet.

```
ps
```

Eine Übersicht zu allen auf dem System laufenden Prozessen lässt sich durch den zusätzlichen Parameter **aux** erreichen:

```
ps aux
```

kill - Prozesse beenden

Prozesse die unerwartet stehen geblieben sind, oder solche die unerwünscht ausgeführt werden, lassen sich mit dem Befehl **kill** beenden. Bei der Verwendung des Befehls ist es nötig neben optionalen Parametern die **PID** (**P**rocess **Id**) des jeweiligen Prozesses anzugeben.

```
kill [Optionen] [PID]
```

Standardmäßig sendet der Befehl das Signal **SIGTERM** (15) an den Kernel, um die Ausführung des jeweiligen Prozesses abzuschließen und ihn anschließend zu beenden. Mithilfe der **Option -s** lässt sich auch explizit festlegen, welches Signal an den Kernel gesendet werden soll:

```
kill -s [Signal] [PID]
```

Dies ist besonders dann hilfreich, wenn sich ein besonders hartnäckiger Prozess nicht durch das Standardsignal beenden lässt. In solchen Fällen kann der Kernel beispielweise mit einem **SIGKILL** (9) beauftragt werden, den Prozess hart abzubrechen:

```
kill -s SIGKILL [PID]
```

Beachten Sie jedoch, dass letztere Option immer nur dann eingesetzt werden sollte, wenn das Standardsignal nicht zum Ziel führt, da der Prozess auf diesem Weg nicht sauber beendet wird.

jobs - Status von Shell-Prozessen anzeigen

Mit dem Befehl **jobs** können Sie den Status aller in der Shell gestarteten Prozesse einsehen:

```
jobs
```

Der Befehl wird vor allem dann benötigt, wenn Sie lang laufende Prozesse im Hintergrund ausführen möchten. Ähnlich wie bei den beiden Befehlen **top** und **ps** wird neben der **Jobnummer** der aktuelle **Prozessstatus** sowie auch der komplette **Aufrufbefehl** angezeigt. Die Ausgabe sieht dann etwa folgendermaßen aus:

```
[3]+  Angehalten              wget -nv https://profi-tutorials.de/it-consulting/
[4]-  Fertig                  wget -nv https://profi-tutorials.de/e-business/
root@hellix:/home/hellberg#
```

& - Prozesse im Hintergrund starten

Wenn Sie einen Prozess **im Hintergrund starten** möchten, dann können Sie dem jeweiligen Befehl ganz einfach ein **Kaufmanns-Und &** anhängen:

```
[Befehl] [Optionen] &
```

Nach Eingabe wird im Terminal die zugewiesene **Jobnummer** und die **PID** ausgegeben.

bg - Vordergrundprozess in den Hintergrund verlagern

Auch bereits im Vordergrund gestartete Prozesse lassen sich nachträglich in den Hintergrund verlagern. Mit der Tastenkombination **[Ctrl+z]** muss der laufende Prozess dazu zunächst angehalten werden. Anschließend kann er mit dem folgenden Befehl im Hintergund fortgeführt werden:

```
bg %[Jobnummer]
```

fg - Hintergrundprozess in den Vordergrund verlagern

Hintergrundprozesse können umgekehrt auch in den Vordergrund verlagert werden. Verwenden Sie dazu den Befehl **fg**:

```
fg %[Jobnummer]
```

Runlevel

Diverse sogenannte **Runlevel** kontrollieren unter Linux-/UNIX-Betriebssystemen, welche Prozesse und Services automatisch - beispielsweise beim Systemstart - mitgestartet werden sollen. In diesem Abschnitt lernen Sie den Befehl **init**, die 7 verfügbaren Runevel, sowie den Standard-Runlevel kennen.

Beachten Sie, dass die Runlevel in der hier beschriebenen Form nur bei dem **init-System** bzw. Paket **sysvinit** zum Einsatz kommen. Das **init-System** wurde unter Debian seit der Version 8 durch **systemd** ersetzt.

init - Runlevel wechseln

Der Befehl **init** wird eingesetzt, um den Runlevel des Systems zur Laufzeit zu wechseln. Die Syntax des Befehls lautet ganz einfach:

```
init [Runlevel]
```

Die verfügbaren Runlevel, welche sich mit dem init-Befehl einstellen lassen, werden nun einmal kurz beschrieben:

Die Runlevel sind durchnummiert von 0 bis 6 und führen jeweils zu unterschiedlichen Betriebsarten des Systems: Der **Runlevel 0** führt zu einem **Systemhalt**, kann also eingesetzt werden, um das System zu stoppen. Der **Runlevel 1** startet den **Einzel-Nutzerbetrieb** des Systems. Es gibt dabei keine Netzwerkverbindung. **Runlevel 2** startet den **lokalen Multi-Nutzerbetrieb**, wobei auch hier wieder die Netzwerkverbindung deaktiviert bleibt. Der **dritte Runlevel (3)** führt zum Start des **vollen Multi-Nutzerbetriebs** - hierbei werden sämtliche Services, wie beispielsweise ein Apache Webserver oder Mailserver, ebenfalls gestartet und es besteht eine Netzwerkverbindung. Ähnlich funtioniert auch **Runlevel 5**, jedoch wird bei diesem auch die **grafische Benutzeroberfläche** (KDM / XDM / GDM) gestartet. Schließlich kann das System mithilfe des letzten **Runlevel (6) neugestartet** werden.

Der **vierte Runlevel** wurde absichtlich übersprungen, da dieser noch **nicht belegt** ist.

Standard-Runlevel

Der **Standard-Runlevel**, welcher bei jedem Systemstart eingesetzt wird, wird in der Datei **/etc/inittab** durch den Eintrag **initdefault** festgelegt. In den meisten Fällen wird hier der volle Multi-Nutzerbetrieb ohne grafische Benutzeroberfläche **(3)**, oder der entsprechende Multi-Nutzerbetrieb mit aktivierter grafischer Benutzeroberfläche **(5)** verwendet.

Ein-/Aushängen mit mount

In diesem kurzen Abschnitt soll der Befehl **mount**, welcher unter UNIX-/Linux-Betriebssystemen zum **Einbinden und Aushängen** von Dateisystemen verwendet wird, vorgestellt.

Dateisysteme anzeigen

Zunächst können Sie durch Eingabe des Befehls **mount** ohne jegliche Optionen alle momentan in das System eingebundenen Dateisysteme ausgeben:

```
mount
```

Die Ausgabe erfolgt zeilenweise - Am Anfang jeder Zeile steht dabei die **Bezeichnung** des Dateisystems oder Geräts. Darauf folgt der **Pfad** auf welchem das Dateisystem eingehängt wurde. Zum Schluss folgen Informationen zum **Typ** (type) des Dateisystems, beispielsweise **ext4**, sowie zu verschiedenen festgelegten **Einhängeoptionen**, wie beispielsweise der **GID**, **UID** oder der **Dateinamen Enkodierung**.

Mit der **Option -l** lässt sich am Ende jeder Zeile zusätzlich auch noch ein **Label** in eckigen Klammern darstellen:

```
mount -l
```

Dateisystem einhängen

Je nachdem, ob es in der sogenannten **/etc/fstab** bereits einen Eintrag gibt, werden unterschiedliche Syntaxen für den **mount** Befehl benötigt. Die genaue Funktionsweise der /etc/fstab wird in einem Beitrag behandelt - dementsprechend wird hier die Einhängung ohne vorhandene Eintragung beschrieben.

Dem Befehl müssen neben **Optionen**, das **Gerät bzw. Dateisystem** sowie der **Einhängepunkt (Mountpoint)** übergeben werden. Außerdem werden **root-Rechte** benötigt. Die Syntax lautet damit:

```
mount [Optionen] [Gerät / Dateisystem] [Mountpoint]
```

Für das **temporäre** Einhängen ohne Eintragung in der **/etc/fstab** können Sie unter **[Gerät / Dateisystem]** eine sogenannte **Gerätedatei (device)** angeben, welche Sie unter UNIX-/Linux-Betriebssystemen im Verzeichnis **/dev/** finden. Beispielsweise sind /dev/sda1 und /dev/sda2 die jeweils erste Partition und zweite Partition eines SCSI Device.

Als **Mountpoint** dient ein beliebiger Pfad eines **Verzeichnisses**, auf dem Sie das neue Gerät einhängen möchten. Das spezifizierte Verzeichnis sollte vorzugsweise leer sein, da jegliche Inhalte nach der Einhängung nicht mehr sichtbar sind - sie werden von dem zweiten Dateisystem **überlagert**, sind jedoch nach dem Aushängen des zweiten Dateisystems wieder verfügbar.

Optionen

Der **mount** Befehl besitzt eine Vielzahl **Optionen**, von denen die wichtigsten einmal kurz in der folgenden Tabelle aufgelistet werden.

Option	Funktion
-a	Es werden alle Dateisysteme der **/etc/fstab** (der angegebenen **Typen**) eingebunden, solange diese nicht mit der **Einhängeoption noauto** versehen wurden. (**a**ll)
-U [UUID]	Es wird das Dateisystem mit der angegebenen **UUID** eingehängt.
-L [Label]	Es wird das Dateisystem mit dem angegebenen **Label** eingehängt.
-B	Ein bereits gemounteter **Teilbaum** kann mit dieser Option an einer anderen Position **erneut eingehängt** werden. Der alte **Mountpoint** bleibt dabei erhalten, sodass die Inhalte an zwei Orten erscheinen. (**B**ind)
-M	Ein bereits gemounteter **Teilbaum** wird an eine andere Position **verschoben**. (**M**ove)

Option	Funktion
-f	Das Einhängen des angegebenen Dateisystems wird lediglich **simuliert**, also nicht tatsächlich durchgeführt. Durch Zusatz der **Option -v** können die Aktionen des **mount** Befehls nachvollzogen werden. (**f**ake)
-o [Parameter]	Wird verwendet, um die **Einhängeoptionen** festzulegen. Als Parameter übergeben Sie hierbei eine durch Kommata getrennte Liste von **Einhängeoptionen**. Verfügbare **Einhängeoptionen** finden Sie im Abschnitt **"Einhängeoptionen"**. (**o**ptions)
-r	Das Dateisystem wird **schreibgeschützt** eingehängt. **Synonym** dieser **Befehlszeilenoption** ist das Hinzufügen der **Einhängeoption ro** durch **-o ro** (read **o**nly)
-w	Das Dateisystem wird **les- und schreibbar** eingehängt. Es handelt sich hierbei um die Voreinstellung bzw. den **Standardwert** des **Kernels**. Auch hier ist **-o rw** ein **Synonym** (**r**ead **w**rite)
-t	Definiert den **Typen** des einzuhängenden Dateisystems. Beispielsweise: ext, ext2, ext3, ext4, nfs, iso9660, vfat (=FAT32), ntfs etc. (**t**ype)
-v	Aktiviert den ausführlichen Modus. (**v**erbose)

Archivierung

Hier sollen Ihnen kurz die Grundfunktionen des Befehls **tar** (**t**ape **a**rchiver), welcher unter Linux-/UNIX-Betriebssystemen für die Erstellung von **.tar Archiven** sowie die **Extraktion** diverser archivierter Dateien verwendet wird,vorgestellt werden. Bei der Systemadministration spielt der Befehl oft eine zentrale Rolle, da er beispielsweise auch bei der Durchführung von Backups eingesetzt werden kann.

Die allgemeine Syntax lautet:

```
tar [Optionen] [Datei(en) / Verzeichnis(se)]
```

Im Folgenden sollen die wichtigsten Optionen zur Erstellung, Auflistung und Extraktion von Archiven kurz genannt werden.

-c - Archive erstellen

Zur **Erstellung** neuer Archive wird die **Option -c** des **tar** Befehls verwendet. Die Syntax lautet:

```
tar -c -f [Archivname] [Datei(en) / Verzeichnis(se)]
```

Benötigt wird hier zusätzlich die vielfach verwendete **Option -f** (**f**ile), welche den Namen des zu Erstellenden Archivs festlegt - bestenfalls endet dieser mit der **Dateiendung .tar**.

-t - Archivinhalt ausgeben

Die **Option -t** wird verwendet, um **Inhalt** eines Archivs **auszugeben**. Grundsätzlich kann auf die Angabe **[Datei(en) / Verzeichnis(se)]** verzichtet werden, wenn ausnahmslos alle Inhalte ausgegeben werden sollen:

```
tar -t -f [Archivname]
```

Wenn Sie jedoch den Inhalt bestimmter Verzeichnisses des Archivs ausgeben möchten, dann fügen Sie die jeweiligen **relativen Pfade** einfach zusätzlich an:

```
tar -t -f [Archivname] [Datei(en) / Verzeichnis(se)]
```

-x - Archive extrahieren

Mit der **Option -x** ist es schließlich möglich, Archive auch wieder zu **extrahieren** - erst dann ist der Zugriff auf die enthaltenen Dateien wieder möglich. Wird bei der Extraktion keine Angabe neben dem Archivnamen gemacht, so wird stets der **gesamte Inhalt** des jeweiligen Archivs in das **aktuelle Arbeitsverzeichnis** extrahiert:

```
tar -x -f [Archivname]
```

Durch Angabe von **Dateien und / oder Verzeichnissen des Archivs** können Sie jedoch auch genau spezifizieren, was extrahiert werden soll:

```
tar -x -f [Archivname] [Datei(en) / Verzeichnis(se)]
```

Schließlich können Sie durch Zusatz der **Option -C** auch ein **Zielverzeichnis** für die Extraktion bestimmen:

```
tar -x -f [Archivname] -C [Zielverzeichnis] [Datei(en) / Verzeichnis(se)]
```

Systemzeit und Hardware-Uhr

Unter Linux/UNIX gibt es prinzipiell zwei unterschiedliche Uhren: Die Systemzeit und die Hardware-Uhr. Während die Systemzeit von Programmen und Anwendungen zur Laufzeit verwendet wird, läuft die Hardware-Uhr zu jeder Zeit - auch wenn das System herunterfahren ist weiter, um nach dem Systemstart wieder die Systemzeit zu synchronisieren.

In diesem Beitrag lernen Sie zunächst die wichtigsten Funktionen des Befehls **date** kennen, welcher zur Ausgabe und Einstellung der Systemzeit verwendet wird. Anschließend wird auch noch auf den Befehl **hwclock** eingegangen - dieser gibt Ihnen die Möglichkeit, den Wert der Hardware-Uhr auszugeben, sowie diverse Einstellungen vorzunehmen.

date - Systemzeit

Ausgabe der Systemzeit

Im Folgenden wird der Befehl **date** mit seinen wichtigsten Optionen vorgestellt. **date** lässt sich ohne Angabe jeglicher Optionen einsetzen und gibt dann die Systemzeit (Datum und Uhrzeit) in einem festen Format aus:

```
date
```

Die Ausgabe der aktuellen Zeit ist durch Anfügen diverser Optionen auch in verschiedenen Formaten möglich. So wird mithilfe der **Option -I** eine Ausgabe nach **ISO 8601** erreicht. Über das Feld **[FMT]** können Sie die Genauigkeit der Ausgabe bestimmen: Erkannt werden **'hours'** für Stunden, **'minutes'** für Minuten, **'seconds'** für Sekunden und **'ns'** für Nanosekunden.

```
date -I[FMT]
```

Weiterhin ist auch die Ausgabe im Format nach **RFC 2822** möglich:

```
date -R
```

Wenn Sie das Format der Ausgabe selbst bestimmen möchten, dann verwenden Sie den **date**-Befehl einfach in Kombination mit verschiedenen **Formatangaben**. Das Format wird in **Anführungszeichen** und mit einem vorangestellten **Plus '+'** angegeben:

```
date "+[Format]"
```

Welche gültigen Formatangaben es gibt, können Sie auch in unserem entsprechenden Beitrag nachlesen.

Setzen der Systemzeit

Mit der **Option -s** kann die Systemzeit auf einen von Ihnen bestimmten Wert gesetzt werden. Spezifiziert wird die Zeit mithilfe eines sogenannten date-String:

```
date -s [String]
```

Das Format des **date**-String kann sehr unterschiedlich aussehen. Hierbei soll nun lediglich ein Format vorgestellt werden, welches für die meisten Anwendungsfälle bestens geeignet ist. Die Syntax des gesamten Befehls lautet dann:

```
date -s "YYYY-MM-DD hh:mm:ss [Zeitzone]"
```

Weitere mögliche String-Angaben finden Sie in unserem oben genannten Beitrag zum **date**-Befehl sowie auf der Info Seite des Befehls.

hwclock - Hardware-Uhr

Ausgabe der aktuellen Zeit

Die Ausgabe der aktuellen Zeit ist auch bei der Hardware-Uhr leicht durchzuführen: Der Befehl **hwclock** wird dazu ohne Angabe von Optionen ausgeführt:

```
hwclock
```

Alternativ kann jedoch auch die **Option -r** verwendet werden, welche dieselbe Ausgabe liefert:

```
hwclock -r
```

Die Ausgabe erfolgt immer nach dem **Systemgebietsschema**, also der **lokalen Zeit**.

Hardware-Uhr einstellen

Zur Einstellung der Hardware-Uhr stehen Ihnen zwei verschiedene Möglichkeiten zur Verfügung. So können Sie die Systemzeit einerseits mithilfe der **Option -w** nach der aktuellen Systemzeit stellen:

```
hwclock -w
```

Andererseits ist es auch möglich, ähnlich wie bei dem **date**-Befehl, die Uhr auf Grundlage einer **Datumszeichenkette** (**date**-String) zu stellen. Verwenden Sie dazu die beiden **Optionen --set** und **--date**:

```
hwclock --set --date='[String]'
```

Das **Format** der **Datumszeichenkette** kann beispielsweise die folgende Form annehmen:

```
hwclock --set --date='YYYY-MM-DD hh:mm:ss'
```

Weiterführende **Optionen** und **Funktionen** zum Befehl **hwclock** finden Sie auch in dem dazugehörigen Beitrag.

Sicheres Löschen von Daten

In dem folgenden Abschnitt wird Ihnen der Befehl **shred** vorgestellt, mit welchem Sie **sicher Daten überschreiben** und anschließend löschen können. Einige der wichtigsten Optionen werden zusätzlich aufgelistet.

Dateien überschreiben

Wenn Sie einzelne Dateien mit Zufallsdaten überschreiben möchten, dann übergeben Sie dem Befehl **shred** einfach die **Pfade** der jeweiligen Dateien. Mehrere Dateien können durch Leerzeichen getrennt hintereinander angegeben werden. Optionen werden nicht zwingend benötigt.

```
shred [Datei(en)]
```

Wenn Sie Dateien nach dem Überschreiben auch noch Löschen möchten, dann verwenden Sie zusätzlich die **Option -u**:

```
shred -u [Datei(en)]
```

Geräte überschreiben

Durch Angabe eines **Gerätenamen** können Sie auch ganze **Blockdevices**, wie beispielsweise Partitionen oder Festplatten überschreiben. Auch hier werden wieder keine Optionen benötigt.

```
shred [Gerät]
```

Durch Hinzufügen der **Option -f** ist es möglich, den Vorgang des Überschreibens zu **erzwingen** - die Option ändert Zugriffsberechtigungen automatisch, wenn dies für die Ausführung des Befehls nötig ist. Eingesetzt werden kann die Option bei Verwendung von **shred** zum Überschreiben von Dateien sowie Geräten.

```
shred -f [Gerät / Datei(en)]
```

Wenn Sie zusätzlich festlegen möchten, wie oft die angegebenen Daten überschrieben werden, dann können Sie mit der **Option -n** einfach einen entsprechenden numerischen Wert an den Befehl übergeben:

```
shred -n [N] [Gerät / Datei(en)]
```

Standardmäßig werden Dateigrößen von **shred** auf die nächste volle Blockgröße aufgerundet - dieses Verhalten lässt sich mithilfe der **Option -x** jedoch auch ausschalten:

```
shred -x [Gerät / Datei(en)]
```

Zur Anzeige des **Fortschritts** während des Überschreibens geben Sie die **Option -v** an:

```
shred -v [Gerät / Datei(en)]
```

Hardware-Informationen

In diesem Abschnitt lernen Sie einige nützliche Befehle kennen, welche das Auslesen von Hardware-Informationen auf Linux-/UNIX-Betriebssystemen ermöglichen.

Das im Folgenden mehrfach benötigte Paket **lshw**, welches nicht auf allen Systemen standardmäßig vorinstalliert ist, können Sie mit dem folgenden Befehl nachinstallieren:

```
apt-get install lshw
```

Überblick zur Hardware-Konfiguration

Den Befehl **lshw** können Sie verwenden, um einen detaillierten Überblick zur gesamten **Hardware-Konfiguration** eines Systems zu erhalten. Führen Sie ihn dazu ohne jegliche Optionen und Parameter aus:

```
lshw
```

Zur Ausgabe gehören unter anderem Informationen zu **Speicher-Konfiguration**, **Firmware-Versionen**, **Mainboard-Konfiguration**, **CPU**, **Cache** und **Bus-Geschwindigkeiten**.

CPU-Informationen

Mit dem Befehl **lscpu** können Sie nähere Informationen zur **CPU** des Systems ausgeben. Es werden verschiedenste Informationen, wie beispielsweise **Architektur**, **Anzahl Kerne**, **Threads**, **Hersteller-ID**, **Modellname** und **Frequenz**, ausgegeben.

```
lscpu
```

Den schon bekannten Befehl **lshw** können Sie zusammen mit der **Option -C** und dem Parameter **'cpu'** verwenden, um eine ähnliche, jedoch etwas unübersichtliche, Ausgabe zu erhalten:

```
lshw -C cpu
```

RAM-Informationen

Informationen zum **Arbeitsspeicher** lassen sich ebenfalls mit **lshw** ausgeben. Verwenden Sie hierbei die **Option -C** mit dem **Parameter 'memory'**. In der Ausgabe werden für jeden verbauten **DIMM** einzeln der **Typ** sowie die **Kapazität**, **Geschwindigkeit** und **Spannung** ausgegeben.

```
lshw -C memory
```

Laufwerksinformationen

Wird der Befehl **lshw** mit der **Option -C** und dem **Parameter 'disk'** ausgeführt, so gibt er Informationen zu allen **Laufwerken** des Systems aus:

```
lshw -C disk
```

Informationen zu USB- und PCI-Geräten

Weiterhin können Informationen zu **USB-** und **PCI-Bussen** eines Systems ausgegeben werden. Verwenden Sie dazu die Befehle **lsusb** und **lspci**.

Zur Ausgabe aller per **USB** an das System angeschlossenen Geräte führen Sie den Befehl **lsusb** einfach ohne jegliche Optionen und Parameter aus. Die Ausgabe erfolgt dann zeilenweise, wobei der **Bus**, die **Hardware-ID** und ein **Namensteil** aus **Vendor-** und **Product-ID** enthalten sind.

```
lsusb
```

Mit **lspci** werden Ihnen Informationen zu den **PCI-Bussen** und angeschlossenen Geräten angezeigt:

```
lspci
```

Informationen zu Netzwerkgeräten

Zur Anzeige von Informationen zu **Netzwerkschnittstellen**, wie beispielsweise der **Bezeichnung**, dem **Herstellernamen**, **Bus**, **Kapazitäten** und **Frequenzen**, verwenden Sie den Befehl **lshw** mit der **Option -C** und dem **Parameter 'network'**:

```
lshw -C network
```

Festplattenpartitionierung

In diesem Abschnitt zur Festplattenpartitionierung lernen Sie den Befehl **fdisk** kennen, welcher zum **Erstellen**, **Anzeigen**, **Bearbeiten** und **Löschen** von **Partitionen** verwendet werden kann. Das Werkzeug unterstützt eine Vielzahl von Partitionstypen, wie beispielsweise **DOS**, **Linux**, **FAT32** oder **NTFS** - **GPT-Partitionstabellen** werden dagegen nicht unterstützt.

Da **fdisk** Teil des Standard-Softwarepakets **utils-linux-ng** ist, muss es bei den meisten Linux Distributionen nicht nachinstalliert werden.

Partitionstabellen auflisten

Mit der **Option -l** des Befehls ist es möglich, die **Partitionstabelle** von Geräten auszugeben. Die Ausgabe aller unter **/proc/partitions** aufgelisteten Geräte mitsamt ihrer Partitionstabelle erfolgt bei Verwendung von **fdisk** und der **Option -l** ohne jegliche zusätzliche Parameter:

```
fdisk -l
```

Wird jedoch ein **Gerät** als Parameter angegeben, so gibt **fdisk** die Partitionstabelle für genau dieses Gerät aus:

```
fdisk -l [Gerät]
```

Partitionstabellen bearbeiten

Die **Bearbeitung** der **Partitionstabelle** eines Geräts ist mithilfe des menügesteuerten Bereichs von **fdisk** möglich. Dieser ist durch Eingabe von **fdisk** mit dem jeweiligen **Gerätenamen** möglich:

```
fdisk [Gerät]
```

Im folgenden werden die wichtigsten **Kommandos** für die **Bearbeitung** von **Partitionstabellen** genannt - einen vollständigen Überblick zu allen verfügbaren Kommandos können Sie sich durch Eingabe von **'m'** ausgeben lassen.

p - Print

Mit dem **Kommando 'p'** lässt sich die **Partitionstabelle** des aktuellen Geräts ausgeben. Die Ausgabe erfolgt ähnlich wie auch bei der direkten Verwendung der **Option -l** mit **fdisk**.

o - Overwrite

Eine **neue leere DOS**- bzw. **MBR-Partitionstabelle** lässt sich mit dem **Kommando 'o'** anlegen. Beachten Sie, dass die neue Partitionstabelle eine bereits existierende Tabelle bei Bestätigung der Änderung **überschreibt**.

n - New

Das **Kommando 'n'** wird verwendet, um **neue Partitionen** auf dem aktuellen Gerät zu erstellen. Ein Assistent führt Sie dabei durch die erforderlichen Schritte.

d - Delete

Eine **Partition** kann mithilfe von **'d'** wieder aus der Partitionstabelle eines Geräts **gelöscht** werden.

a - Active

Bei **DOS**- bzw. **MBR-Partitionstabellen** lässt sich das **Kommando 'a'** verwenden, um das **Boot-Flag** einer Partition zu setzen.

w - Write

Die **Änderungen** an der **Partitionstabelle** eines Geräts werden von **fdisk** erst bei Verwendung des **Kommandos 'w'** geschrieben und damit **wirksam**. Nach Eingabe des Kommandos werden alle Änderungen auf das Gerät geschrieben und **fdisk** beendet.

Aliase definieren / löschen

Mit dem Befehl **alias** lassen sich alternative Aufrufnamen (Aliase) für Befehlsaufrufe definieren. In diesem Abschnitt soll der Befehl in seiner grundlegenden Anwendung vorgestellt werden.

Alias definieren

Die allgemeine Syntax zur **Definition eines neuen Alias** lautet folgendermaßen:

```
alias [Aliasname]='[Befehl]'
```

Der auf diese Weise definierte Alias wird mit **[Aliasname]** aufgerufen und führt dann den Befehl **[Befehl]** aus.

Aliase anzeigen

Ein mit **alias** erzeugter Alias kann auch **ausgegeben** werden, um Einsicht in den für ihn festgelegten Befehl zu erlangen. Dazu wird dem Befehl **alias** einfach der jeweilige **Aliasname** übergeben:

```
alias [Aliasname]
```

Die **Ausgabe** dieses Befehls hat dann die Form: **[Aliasname]='[Befehl]'**. Schließlich ist es auch möglich **alle** bereits definierten Aliase auszugeben. Hierzu wird der Befehl **alias** ohne jegliche Parameter oder Optionen verwendet:

```
alias
```

Alias löschen

Zwar existieren so gesetzte Aliase nur bis zur **Beendigung der Shell**, in welcher sie definiert wurden, so ist es dennoch möglich sie auch ausdrücklich zu **löschen**. Hierzu wird der Befehl **unalias** eingesetzt:

```
unalias [Aliasname]
```

Wie Sie einen **dauerhaften Alias definieren**, erfahren Sie in einem dedizierten Beitrag.

Datei-/Verzeichnis-Metadaten

Die **Ausgabe der Metadaten** - etwa Zeitstempel, Berechtigungen, Besitzer / Gruppe, und Datentyp von **Dateien** sowie auch **Verzeichnissen** des Dateisystems ist mit dem Befehl **stat** durchzuführen. Der Befehl soll in diesem Abschnitt einmal kurz in seinen Grundfunktionen vorgestellt werden.

stat lässt sich ohne Angabe jeglicher Optionen auf einer Datei bzw. einem Verzeichnis ausführen, um eine **ausführliche Ausgabe** der Metadaten zu erreichen. Die Ausgabe setzt sich in diesem Fall aus Informationen zu Name, Größe, Blöcke, EA Block, Typ, Gerät, Inode, Verknüpfungen, Berechtigungen sowie Zugriffs- / Modifizierungs- / Änderungs- und Erstellungszeitstempel zusammen.

```
stat [Datei / Verzeichnis]
```

Eine **Ausgabe in Kurzform** ist mithilfe der **Option -t** möglich - hierbei erfolgt die Ausgabe dann auf einer Zeile:

```
stat -t [Datei / Verzeichnis]
```

Unter Verwendung der **Option -t** werden die einzelnen Daten durch Leerzeichen getrennt und ohne jegliche Beschriftungen in folgender Reihenfolge ausgegeben: Dateiname, Gesamtgröße (in Bytes), Anzahl reservierter Blöcke, Raw-Modus, Benutzer-ID (Besitzer), Gruppen-ID, Gerätenummer, Inode-Nummer, Anzahl harter Verknüpfungen, Major-Gerätetyp, Minor-Gerätetyp, Letzte Zugriffszeit, Letzte Daten-Modifizierungszeit, Letzte Status-Änderungszeit, Erstellungszeit und schließlich ein Hinweis auf die optimale I/O-Übertragungsgröße.

Das Format der Ausgabe von **stat** kann mithilfe der **Option -c** und speziellen **Formatangaben** auch selbst spezifiziert werden:

```
stat -c [Format] [Datei / Verzeichnis]
```

Die **Option --printf** wird ebenfalls zur Angabe eines eigenen Formats verwendet, erlaubt im Vergleich zur **Option -c** jedoch auch **Maskierungen** mit dem **Rückschrägstrich "\"** - etwa **"\n"**, um mithilfe von Zeilenumbrüchen eine mehrzeilige Ausgabe zu erzeugen.

```
stat --printf [Format] [Datei / Verzeichnis]
```

Informationen zu den bei den **Optionen -c** und **--printf** verwendeten Formatangaben, den jeweils ausgegebenen Informationen sowie weiteren verfügbaren Optionen finden Sie in unserem dedizierten Beitrag zu diesem Thema.

Extended Arguments

Der in diesem Abschnitt beschriebene Befehl **xargs** (**ex**tended **arg**ument**s**) kann in verschiedenen Situationen eingesetzt werden, um etwa große Argumentlisten mit einem beliebigen Befehl zu verarbeiten, oder sogar um eine parallelisierte Abarbeitung zugunsten einer höheren Performance zu erreichen.

Es sollen hier nur die wichtigsten Anwendungen von **xargs** vorgestellt werden. Für nähere Informationen zu den in **xargs** verfügbaren Optionen sowie verschiedene praktische Beispiele, lesen Sie auch unseren Beitrag zu diesem Thema.

Der Befehl **xargs** lässt sich allgemein ohne jegliche Optionen einsetzen, um eine vom Standard-Input **stdin** eingelesene Liste von Argumenten in eine oder mehrere Kommandozeilen umzuwandeln und diese auszuführen. Standardmäßig werden dabei so viele Argumente wie möglich an das Ende des **xargs'** übergebenen Befehls sowie seiner initialen Argumente gehängt.

```
xargs [Optionen] [Befehl [Initiale Argumente]]
```

Der Standard-Output **stdout** eines Befehls (z.B. **ls**) ließe sich so etwa per Pipe an **xargs** übergeben, um dort weitere Aktionen mit den Argumenten durchzuführen:

```
[Befehl] | xargs [Optionen] [Befehl [Initiale Argumente]]
```

Anzahl Argumente

Möchten Sie lediglich ein Argument pro Kommandozeile verwenden, so können Sie dies mit der **Option -n** und dem Parameter 1 spezifizieren. Jedes Argument erhält so eine eigene Kommandozeile - dies ist besonders dann wichtig, wenn der mit **xargs** verwendete Befehl selbst nicht die Angabe einer Argumentliste unterstützt.

```
xargs -n 1 [Befehl [Initiale Argumente]]
```

Platzhalter ersetzen

Wenn die eingelesenen Argumente nicht an das Ende des Befehls angehängt werden sollen und stattdessen an anderer Stelle eingefügt werden müssen, so ist die Verwendung eines Platzhalters zur Ersetzung möglich. Der Platzhalter wird mit der **Option -I** angegeben und vor der Ausführung der Kommandozeilen jeweils durch die Argumente ersetzt. Diese Option findet etwa bei dem Kopieren vieler einzelner Dateien in ein bestimmtes Zielverzeichnis mit dem Befehl **cp** Anwendung.

In diesem Fall wird die Zeichenkette **"{}"** zur Ersetzung verwendet und durch die **Option -n** mit dem Parameter 1 jeweils ein Argument eingesetzt:

```
xargs -I {} [Befehl [Initiale Argumente]] {} [...]
```

Parallelisierung

Schließlich kann **xargs** die Ausführung von Befehlen bzw. die Abarbeitung von Aufgaben parallelisieren: Hierbei kommt die **Option -P** zum Einsatz, welche eine maximale Anzahl Prozesse definiert, die **xargs** zu einem Zeitpunkt gleichzeitig ausführen darf. Anwendung findet diese Option etwa zur Parallelisierung von **rsync** bei der Übertragung vieler kleiner Dateien zwischen zwei Systemen, oder dem Download vieler kleiner Dateien aus dem Internet mit **wget**.

In vielen Fällen ist es sinnvoll hier die maximale Anzahl Argumente pro ausgeführter Kommandozeile bzw. ausgeführtem Prozess zu beschränken - unter Umständen wird andernfalls nur ein Prozess mit allen Argumenten auf einer Kommandozeile gestartet. Im untenstehenden Befehl wird daher die **Option -n** zusätzlich eingesetzt:

```
xargs -n 1 -P [Anzahl Prozesse] [Befehl [Initiale Argumente]]
```

Netzwerke

Konfiguration und Statistiken

In diesem Abschnitt lernen Sie einige der wichtigsten Befehle für die Konfiguration von Netzwerken sowie für die Ausgabe relevanter Informationen bezüglich Erreichbarkeit etc. kennen.

ip - Befehl für die Netzwerkkonfiguration

Der Befehl **ip** ist Teil des Pakets **iproute2** und ersetzt den **ifconfig**-Befehl der **net-tools**.

```
ip [Optionen]
```

Der Befehl kann mit vielen verschiedenen Optionen ausgeführt werden, um **Netzwerkschnittstellen abzufragen** oder zu **konfigurieren**.

Mit der **Option link** ist es möglich, den **Status aller Schnittstellen** abzufragen:

```
ip link
```

Wollen Sie den **Status der Schnittstellen** genauso abfragen, wie dies mit **ifconfig** möglich war, fügen Sie einfach die **Option -s** zusätzlich mit an:

```
ip -s link
```

Weiterhin können mit **link Einstellungen** auf **Ethernet-Ebene** geändert werden. Mit dem folgenden Befehl können Sie die **MAC-Adresse ändern**:

```
ip link set dev [Gerät] address [MAC-Adresse]
```

Außerdem können Sie eine **Schnittstelle aktivieren (up)** bzw. **deaktivieren (down)**:

```
ip link set [up/down] [Gerät]
```

Mit der Option **addr** können Sie alle verfügbaren **Netzwerkschnittstellen** mitsamt ihrer **IP-Adressen abfragen**, anzeigen und neue **IP-Adressen hinzufügen**. Mit **addr show** werden Ihnen die Daten übersichtlich ausgegeben:

```
ip addr show
```

Zudem können Sie mit **addr add** IP-Adressen **zuweisen**:

```
ip addr add [IP-Adresse] dev [Gerät]
```

Mit **del** können Sie eine IP-Adresse wieder **entfernen**:

```
ip addr del [IP-Adresse] dev [Gerät]
```

ping - Erreichbarkeit überprüfen

Mit dem Befehl **ping** können Sie die **Erreichbarkeit** einer Netzwerkschnittstelle überprüfen. Die Syntax lautet:

```
ping [Optionen] [Adresse]
```

Der Befehl sendet bei Ausführung Anfragen an die "angepingte" Netzwerkschnittstelle, welche diese bei Eingang beantwortet. Mit der Tastenkombination **[Ctrl+c]** können Sie die Ausführung des Befehls wieder beenden.

nslookup - DNS-Abfragen durchführen

Der Befehl **nslookup** wird verwendet, um **DNS-Abfragen** durchzuführen. Das Werkzeug ist nicht unter allen Linux-Distributionen standardmäßig vorinstalliert. Unter Debian erfolgt die **Installation** des hierfür benötigten **Pakets "dnsutils"** folgendermaßen:

```
apt-get install dnsutils
```

nslookup kann sowohl in einem **interaktiven**, als auch in einem **nicht-interaktiven Modus** verwendet werden. Im Folgenden wird der **nicht-interaktive Modus** eingesetzt:

Zum **Auflösen der IP-Adresse eines Domainnamen** wird der folgende Befehl verwendet:

```
nslookup [Domainname]
```

Eine sogenannte **Reverse-Lookup-Abfrage** ist dagegen durch Angabe einer **IP-Adresse** möglich:

```
nslookup [IP-Adresse]
```

Der **MX-Eintrag** einer Domain wird durch Hinzufügen der **Option -query** mit dem **Parameter "mx"** ausgegeben. Analog können auch der **NS-Eintrag ("ns")**, der **SOA-Eintrag ("soa")**, oder **alle Informationen** zu einer Domain **("any")** angezeigt werden.

```
nslookup -query=mx [Domainname]
```

```
nslookup -query=ns [Domainname]
```

```
nslookup -query=soa [Domainname]
```

```
nslookup -query=any [Domainname]
```

Weitere Informationen zur Anwendung und Funktionsweise des Befehls **nslookup** finden Sie in unserem dedizierten Beitrag zum Thema.

traceroute - Datenpaket verfolgen

traceroute ermöglicht Ihnen die **Nachverfolgung eines Datenpakets** auf allen seinen Zwischenstationen. Dazu werden kleine Datenpakete an den gewünschten Host gesendet. Die Syntax des Befehls lautet folgendermaßen:

```
traceroute [Optionen] [Adresse]
```

Optionen werden nicht benötigt.

ss - Socket Statistiken anzeigen

Mit **ss** können Sie Statistiken zu **PACKET**-, **TCP**-, **UDP**-, **DCCP**-, **RAW**- und **Unix-Domain-Sockets** anzeigen:

```
ss [Optionen]
```

Die verschiedenen verfügbaren Optionen dienen der Anzeige diverser Informationen. Die **Option -l** (**l**istening) wird beispielsweise verwendet, um nur **"horchende" Sockets** anzuzeigen:

```
ss -l
```

Die Option -t (**T**CP) zeigt alle **TCP Verbindungen** an:

```
ss -t
```

-u (**U**DP) zeigt ausschließlich **UDP Verbindungen** an:

```
ss -u
```

Schließlich werden mit der **Option -x** (UNI**X**) nur **UNIX Verbindungen** angezeigt:

```
ss -x
```

SSH

Der in diesem Abschnitt beschriebene Befehl ssh kann auf fast allen Linux-/UNIX-Betriebssystemen für den sicheren Zugriff auf einen SSH-Server oder ein Remote-System eingesetzt werden. Der Befehl wird für die Anmeldung sowie die Ausführung von Befehlen verwendet.

Login

Der Login, welcher zum Verbindungsaufbau mit einer Remote-Maschine erforderlich ist, erfolgt zunächst ganz einfach durch Eingabe des Befehls **ssh** mit der Adresse des jeweiligen Systems:

```
ssh [Adresse]
```

Wenn Sie bei der Anmeldung einen **Benutzer spezifizieren** möchten, dann können Sie den **Benutzernamen** einfach als **Präfix** und durch ein '**@**' von der Adresse getrennt anhängen:

```
ssh [Benutzer]@[Adresse]
```

Auch die Verwendung der **Option -l** ist hier möglich:

```
ssh -l [Benutzer] [Adresse]
```

Es ist zu beachten, dass bei dem ersten Verbindungsaufbau zu einem **SSH Server** eine besondere Abfrage stattfindet, bei welcher der **Server** und ein sogenannter **Host-Schlüssel** von Ihnen bestätigt werden müssen. Beides wird nach Ihrer Bestätigung in eine Liste **bekannter Hosts** unter "**~/.ssh/known_hosts**" abgelegt. Aus diesem Grund müssen Sie diese Bestätigung auch nur bei dem **ersten Verbindungsaufbau** durchführen - wird der Schlüssel ein weiteres Mal abgefragt, sollten Sie sicher sein, dass der Schlüssel serverseitig entsprechend geändert wurde.

Befehle ausführen

Wenn Sie einen **Befehl** auf einer Remote-Maschine ausführen möchten, können Sie diesen auch direkt dem **ssh** Befehl übergeben:

```
ssh [Adresse] [Befehl]
```

Bei einer solchen Verwendung des **ssh** Befehls wird nach der **Authentifizierung** lediglich der **Befehl** auf der Remote-Maschine ausgeführt und die **Ausgaben in der lokalen Befehlszeile** ausgegeben. Im Anschluss bleibt der Nutzer nicht angemeldet - die **SSH-Sitzung** wird stattdessen sofort nach Befehlsausführung **beendet**.

SCP

Der Befehl **scp** kann auf Linux-/UNIX-Betriebssystemen verwendet, um Dateien über eine sichere, verschlüsselte Netzwerkverbindung zu kopieren. Neben einem **SCP-Client** ist dazu - ähnlich wie bei **SSH** - auch ein entsprechender **Server** notwendig.

Wie der **SCP-Server** installiert wird, können Sie in unserem dedizierten Beitrag zu **SCP** nachlesen - Hier soll lediglich die Syntax von **scp** selbst und seine **wichtigsten Optionen** gezeigt werden.

Dateien kopieren

Zum **Kopieren von Dateien** mit dem Befehl **scp** wird grundsätzlich die folgende Syntax verwendet:

```
scp [Optionen] [Pfad Quelle] [Pfad Ziel]
```

Bei den **Pfadangaben** sind im Falle **lokaler Pfade** keine Besonderheiten zu beachten. Lediglich bei **Pfadangaben** auf **Remote-Maschinen** muss eine spezielle Syntax berücksichtigt werden. Der jeweilige **Hostname** wird durch einen **Doppelpunkt ':'** getrennt links an den Pfad angehängt. Die Angabe eines **Benutzernamen** ist **optional** ebenfalls möglich, indem dieser mit einem **'@'** getrennt links an den **Hostnamen** angehängt wird. SCP ermöglicht zudem auch das Kopieren zwischen zwei Remote-Hosts. Die resultierende Syntax lautet:

scp [Optionen] [Benutzer]@[Host 1]:[Pfad] [Benutzer]@[Host 2]:[Pfad]

Im Folgenden werden einige wichtige Optionen des Befehls **scp** kurz beschrieben.

-3 - Lokaler Rechner als Zwischenstation

Mit der **Option -3** wird die Standardeinstellung, dass Daten beim Kopieren zwischen **zwei Hosts direkt** zwischen diesen versendet werden, überschrieben. Der Kopiervorgang findet somit **über den lokalen Rechner** statt.

```
scp -3 [Benutzer]@[Host 1]:[Pfad] [Benutzer]@[Host 2]:[Pfad]
```

-l - Bandbreite limitieren

Die **Bandbreite** einer Datenübertragung mit dem Befehl **scp** kann mithilfe der **Option -l** limitiert werden. Die Angabe erfolgt in **KBit/s**:

scp -l [Benutzer]@[Host 1]:[Pfad] [Benutzer]@[Host 2]:[Pfad]

-P - Port festlegen

Der verwendete Port, welcher bei **scp** standardmäßig **Port 22** ist, kann mit der **Option -P** beliebig spezifiziert werden:

scp -P [Portnummer] [Benutzer]@[Host 1]:[Pfad] [Benutzer]@[Host 2]:[Pfad]

-r - Verzeichnisse rekursiv kopieren

Zum **rekursiven Kopieren** von Verzeichnissen, verwenden Sie die **Option -r**:

scp -r [Benutzer]@[Host 1]:[Pfad] [Benutzer]@[Host 2]:[Pfad]

-v - Debugging

Die **Option -v** kann schließlich bei der **Fehlerbehebung** helfen - sie sorgt dafür, dass jegliche auftretende **Debugging-Nachrichten** in die **Ausgabe** geschrieben werden:

scp -v [Benutzer]@[Host 1]:[Pfad] [Benutzer]@[Host 2]:[Pfad]

FTP- / HTTP-Download

Für den **Download von Dateien** direkt von **FTP**- oder **HTTP** bzw. **HTTPS-Servern** wird Ihnen in diesem Abschnitt der Befehl **wget** vorgestellt. Der Befehl kann neben der einfachen Verwendung als Downloadmanager auch in Shell-Skripten verwendet, oder ohne Benutzerinteraktion im Hintergrund ausgeführt werden.

Die allgemeine Syntax für das Herunterladen von Dateien lautet folgendermaßen:

```
wget [Optionen] [URL(s)]
```

Wird der Befehl ohne zusätzliche Optionen verwendet, so werden die angegebenen URLs heruntergeladen und im aktuellen Arbeitsverzeichnis abgelegt.

Einige wichtige Optionen werden im Folgenden kurz vorgestellt.

-t - Anzahl Versuche spezifizieren

Kommt es während des Herunterladens zu **Verbindungsabbrüchen**, versucht **wget** standardmäßig bis zu **20** Mal, die Verbindung wiederherzustellen. Wird diese maximale Anzahl der Versuche überschritten, so wird der Vorgang abgebrochen.

Mit der **Option -t** können Sie genau festlegen, wie oft **wget** versuchen soll, die Verbindung wiederherzustellen:

```
wget -t [Anzahl Versuche] [URL(s)]
```

-c - Download fortsetzen

Wurde ein Download aus jeglichen Gründen **abgebrochen**, und sind die **lokalen Dateien** des ersten Vorgangs sind noch auf dem Dateisystem vorhanden, so kann der **Download** mit der **Option -c fortgesetzt** werden:

```
wget -c [URL(s)]
```

-i - URLs aus Datei auslesen

Neben der direkten Eingabe von URLs als Befehlsparameter unterstützt **wget** auch das **Auslesen von URLs** aus einer **lokalen** oder **externen Datei**:

```
wget -i [Dateipfad] [URL(s)]
```

Die URLs werden in der Datei untereinander aufgelistet, sodass für **jede URL eine Zeile** verwendet wird. Der Option kann auch der **Parameter "-"** übergeben werden, um die URLs aus dem **Standard-Input** (stdin) zu lesen. Die Angabe von URLs direkt in der Befehlszeile ist gleichzeitig möglich - in diesem Fall werden zunächst die URLs der Befehlszeile und anschließend die in der übergebenen Datei befindlichen URLs heruntergeladen.

-O - Ausgabedatei spezifizieren

Bei Angabe einer **Ausgabedatei** mithilfe der **Option -O** werden die Inhalte der heruntergeladenen Dateien miteinander **konkateniert** und anschließend in eine **gemeinsame Datei** mit dem von Ihnen festgelegten **Namen/Pfad** geschrieben:

```
wget -O [Dateipfad] [URLs]
```

Durch Angabe des **Parameters "-"** ist es möglich, die Inhalte im **Standard-Output** (stdout) auszugeben.

-P - Ausgabeverzeichnis spezifizieren

Mithilfe der **Option -P** lässt sich das **Verzeichnis-Präfix** auf den angegebenen **Pfad** festlegen, um alle heruntergeladenen Dateien unter diesem abzulegen.

```
wget -P [Verzeichnispfad] [URLs]
```

Der **Standardwert "."** gibt das **aktuelle Arbeitsverzeichnis** an.

-b - Hintergrundausführung

Schließlich kann **wget** mithilfe der **Option -b** im **Hintergrund** ausgeführt werden. Nach Ausführung wird die Befehlszeile wieder freigegeben und die Ausführung anderer Aufgaben ist möglich. Alle erzeugten **Ausgaben** werden in die sogenannte **wget-Log umgeleitet**.

```
wget -b [URL(s)]
```

Related Terms:

- Term: Betriebssystem
- Term: Syntax
- Term: Root-Benutzer
- Term: SSH
- Term: Rekursion
- Term: Wildcard
- Term: Systemd
- Term: Dateisystem
- Term: RFC
- Term: String
- Term: Cache
- Term: Thread
- Term: DNS
- Term: UDP
- Term: Authentifizierung
- Term: FTP
- Term: Konkatenation
- Term: TTY
- Term: ISO
- Term: TCP
- Term: SCP