SelfLinux-0.13.0

Dateien unter Linux

Autor: Frank Boerner (*frank@frank-boerner.de*)
Formatierung: Matthias Hagedorn (*matthias.hagedorn@selflinux.org*)

Im Unterschied zu Windows wird bei Linux zwischen Groß- und Kleinschreibung unterschieden. Daher ist die Datei test nicht identisch mit der Datei Test. Was am Anfang bei Umsteigern womöglich zur Verwirrung beiträgt, erweist sich nach einer Einarbeitungszeit häufig als eine praktische Eigenschaft.

Ein Dateiname darf 255 Zeichen lang sein. Damit sollte es jedem gelingen, seine Dokumente aussagekräftig zu benennen. Wie die Erfahrung zeigt, sind Dateinamen in der Praxis selten auch nur annähernd so lang.

Eine Datei hat unter Linux keine bestimmte Endung, wie dies bei Windows der Fall ist (beispiel.exe oder beispiel.txt). Auch dies verwirrt am Anfang den Umsteiger, doch gewöhnt man sich auch daran.

Es folgen nun eine Reihe von Kommandos, die man im Umgang mit Dateien benötigt.

Inhaltsverzeichnis

1 touch

Das Kommando `touch` legt eine neue Datei mit der Größe 0 an, sofern noch keine Datei gleichen Namens existiert. Existiert die Datei schon, so ändert `touch` das Datum der letzten Änderung.

```
user@linux ~/testdir/ $ ls
user@linux ~/testdir/ $ touch test1
user@linux ~/testdir/ $ ls -l

insgesamt 0
-rw-r--r--    1 user     users           0 Mai 29 17:34 test1
```

```
user@linux ~/testdir/ $ ls -l

insgesamt 4
-r--r--r--    1 user     users         335 Feb 10  2001 profile
-rw-r--r--    1 user     users           0 Mai 29 17:34 test1

user@linux ~/testdir/ $ touch profile
user@linux ~/testdir/ $ ls -l

insgesamt 4
-r--r--r--    1 user     users         335 Mai 29 17:36 profile
-rw-r--r--    1 user     users           0 Mai 29 17:34 test1
```

2 cat

Das Kommando `cat` liest eine oder mehrere Dateien und gibt diese auf der Standardausgabe aus. Die Standardausgabe ist normalerweise das (evtl. virtuelle) Terminal, in welchem sich auch die aktuelle Shell befindet.

```
user@linux ~/testdir/ $ ls

test1

user@linux ~/testdir/ $ cat test1

Dies ist Testdatei 1.
```

`cat` kann aber auch Daten von der Standardeingabe (normalerweise ist dies die Tastatur) lesen und diese in eine Datei schreiben. Das Zeichen `>` bewirkt eine **Umlenkung des Standardausgabekanals** und schreibt alles, was man über die Tastatur eingibt, in die Datei test2.

```
user@linux ~/testdir/ $ cat > test2

Dies ist Testdatei 2.
```

(Die Eingabe wird mit ^D beendet)

```
user@linux ~/testdir/ $ cat test2

Dies ist Testdatei 2.
```

Man kann den Operator > aber auch dazu verwenden, mehrere Dateien in eine einzige umzuleiten. Die beiden Zeilen, die von cat ausgegeben werden, erscheinen so nicht auf dem Bildschirm, sondern werden in die Datei test3 geschrieben.

```
user@linux ~/testdir/ $ cat test1 test2 > test3
user@linux ~/testdir/ $ cat test3

Dies ist Testdatei 1.
Dies ist Testdatei 2.
```

3 cp

Das Kommando cp (copy) kopiert Dateien und Verzeichnisse. Wenn nur zwei Dateinamen als Parameter angegeben werden, wird die erstgenannte Datei in die zweite kopiert. Werden mehrere angegeben, nimmt cp an, dass die letzte Angabe der Name eines Verzeichnisses ist und kopiert alle angegebenen Dateien in dieses Verzeichnis, falls es existiert.

cp hat eine Menge Optionen, die in der Manpage zu finden sind. Hier sind die wichtigsten:

cp -i: Vor dem Überschreiben von Zieldateien nachfragen.
cp -d: Symbolische Links als solche kopieren, nicht die Dateien, auf die verwiesen wird.
cp -R: Rekursiv kopieren, d. h. auch Unterverzeichnisse.

Das folgende Beispiel kopiert die Datei test1 nach Datei test5:

```
user@linux ~/testdir/ $ ls

test1

user@linux ~/testdir/ $ cp test1 test5
user@linux ~/testdir/ $ ls

test1 test5
```

Das folgende Beispiel kopiert die Dateien test1 test2 test3 in das Verzeichnis test4:

```
user@linux ~/testdir/ $ ls -lR

.:
insgesamt 12
-rw-r--r--    1 user     users          22 Mai 29 17:56 test1
```

```
-rw-r--r--    1 user     users          22 Mai 29 17:59 test2
-rw-r--r--    1 user     users          44 Mai 29 18:00 test3
drwxr-xr-x    2 user     users          48 Mai 29 18:11 test4

./test4:
insgesamt 0

user@linux ~/testdir/ $ cp test1 test2 test3 test4
user@linux ~/testdir/ $ ls -lR

.:
insgesamt 12
-rw-r--r--    1 user     users          22 Mai 29 17:56 test1
-rw-r--r--    1 user     users          22 Mai 29 17:59 test2
-rw-r--r--    1 user     users          44 Mai 29 18:00 test3
drwxr-xr-x    2 user     users         120 Mai 29 18:11 test4

./test4:
insgesamt 12
-rw-r--r--    1 user     users          22 Mai 29 18:11 test1
-rw-r--r--    1 user     users          22 Mai 29 18:11 test2
-rw-r--r--    1 user     users          44 Mai 29 18:11 test3
```

4 mv

Das Kommando `mv` (`move`) verschiebt Dateien und Verzeichnisse. Für die Behandlung der angegebenen Dateien und Verzeichnisse gilt das bei `cp` gesagte. `mv` wird auch zum Umbenennen von Dateien und Verzeichnissen verwendet.

`mv -i`: Vor dem Überschreiben von Zieldateien nachfragen.

Hierzu ein Beispiel:

```
user@linux ~/testdir/ $ ls

test1

user@linux ~/testdir/ $ mv test1 datei1
user@linux ~/testdir/ $ ls

datei1
```

Die Datei `test1` wurde in `datei1` umbenannt.

5 rm

Das Kommando `rm` (`remove`) löscht eine oder mehrere Dateien. Um Dateien löschen zu können, benötigt man Schreibrechte in dem jeweiligen Verzeichnis. Wenn diese für das aktuelle Verzeichnis fehlen, muss der

Löschvorgang für jede Datei mit `y` oder `n` bestätigt oder abgelehnt werden.

```
user@linux ~/testdir/ $ ls

datei1

user@linux ~/testdir/ $ rm datei1
user@linux ~/testdir/ $ ls
user@linux ~/testdir/ $
```

Mit der Option `-r` löscht `rm` auch Verzeichnisse, selbst wenn sie nicht leer sind.

```
user@linux ~/testdir/ $ ls -lR

.:
insgesamt 0
-rw-r--r--    1 user     users           0 Mai 29 19:34 datei1
-rw-r--r--    1 user     users           0 Mai 29 19:34 datei2
-rw-r--r--    1 user     users           0 Mai 29 19:34 datei3
-rw-r--r--    1 user     users           0 Mai 29 19:34 datei4
drwxr-xr-x    2 user     users         144 Mai 29 19:34 dir1

./dir1:
insgesamt 0
-rw-r--r--    1 user     users           0 Mai 29 19:34 datei1
-rw-r--r--    1 user     users           0 Mai 29 19:34 datei2
-rw-r--r--    1 user     users           0 Mai 29 19:34 datei3
-rw-r--r--    1 user     users           0 Mai 29 19:34 datei4

user@linux ~/testdir/ $ rm -r *
user@linux ~/testdir/ $
```

weitere nützliche Option sind:

`-i`: interaktives Löschen. Jedes Löschen muss mit `y` oder `n` bestätigt oder abgelehnt werden.
`-f`: löscht auch schreibgeschützte Dateien ohne Bestätigung

Achtung: Wenn man im Wurzelverzeichnis `/` ist und als root den Befehl `rm -r *` ausführt, löscht man ALLE Dateien des Systems! Daher ist rekursives Löschen nur mit äußerster Vorsicht anzuwenden!

Eine "gefährliche" Option soll hier nicht unerwähnt bleiben:

`-d`: Löscht Verzeichnisse mittels eines **unlink** Systemaufrufes. Da hierbei die enthaltenen Dateien nicht mitgelöscht werden, ist das Dateisystem hinterher meist inkonsistent. Es wird dann wahrscheinlich eine Dateisystemreparatur notwendig sein.

6 Jokerzeichen

Um Gruppen von Dateien bearbeiten zu können, benötigt man die sogenannten Jokerzeichen. Diese können z.B. verwendet werden, um eine Dateiauswahl zu treffen.

Man unterscheidet folgende Jokerzeichen:

`?`	genau ein beliebiges Zeichen
`*`	beliebig viele Zeichen
`[abc]`	genau eines der angegebenen Zeichen
`[a-d]`	ein Zeichen aus dem angegebenen Bereich
`[!abc]`	keines des angegebenen Zeichen
`*.html`	würde alle html Dateien auswählen
`*buch*`	würde alle Dateien auswählen, in deren Namen "buch" vorkommt

Durch die Angabe mit eckigen Klammern kann die Auswahl weiter eingeschränkt werden.

7 more und less

`more` und `less` sind sogenannte Pager (engl. to page: weiterblättern). Sie geben den Inhalt einer Datei auf dem Bildschirm aus und halten jeweils nach einer Bildschirmseite an. `less` ist das leistungsfähigere Programm und kann alles, was `more` auch kann (engl. "less is more": "Weniger ist mehr.").

Beide Programme können mit der Taste `q` beendet werden. Die `Leertaste` blättert seitenweise und die `Returntaste` zeilenweise vorwärts.

Weitere Informationen zu diesen Programmen findet man in der jeweiligen Manpage.

8 file

Das Kommando `file` stellt fest, um was für einen Dateityp es sich handelt, da unter Linux im Gegensatz zu Windows der Dateityp ja nicht an der Endung zu erkennen ist.

Dazu ein paar Beispiele:

```
user@linux ~/testdir/ $ file test

test: empty

user@linux ~/testdir/ $ ls -l

insgesamt 0
-rw-r--r--    1 user     users           0 Mai 29 20:10 test

user@linux ~/ $ file testdir

testdir: directory

user@linux ~/ $ file "übersicht texte"

übersicht texte: ISO-8859 text

user@linux ~/selflinux/ $ file slcompile

slcompile: Bourne-Again shell script text

user@linux ~/daten1/c++/ $ file BSP4
```

```
BSP4: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), not stripped

user@linux ~/daten1/c++/ $ file BSP4.cpp

BSP4.cpp: ASCII C program text

user@linux ~/daten1/c++/ $ file BSP4.o

BSP4.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not
stripped
```

Diese Daten bezieht das Kommando dabei aus der Datei `/etc/magic`. Ausführliche Information dazu findet sich in `man file` und `man 4 magic`.