Rechte-Management für Dateien und Verzeichnisse

LPI Essentials

Andreas B. Mundt
andreas.mundt@zsl-bw.de

13. Dezember 2022

5.3 Einstellungen für Dateiberechtigungen und Dateieigentum

Gewichtung: 2

Beschreibung: Verstehen und Bearbeiten von Dateiberechtigungen und Eigentumseinstellungen.

Hauptwissensgebiete:

- Datei- und Verzeichnisrechte und Dateieigentümer

Auszugsweise Liste wichtiger Dateien, Begriffe und Hilfsprogramme:

- `ls -l`
- `ls -a`
- `chmod, chown`

Gut zu wissen:

- `chgrp`

Aus- und Überblick

Besitzer, Gruppen, und der Rest

Die Existenz von verschiedenen Benutzern, die in eine Gruppenstruktur eingeordnet sind, erlaubt die Vergabe von Rechten abhängig von Nutzer und Gruppenzugehörigkeit.

Im klassische Unix–Rechtekonzept gehört eine Datei oder ein Verzeichnisses genau:

- einem Benutzer (`user(u)`),
- einer Gruppe (`group(g)`).

Besitzer, Gruppen, und der Rest

Die Existenz von verschiedenen Benutzern, die in eine Gruppenstruktur eingeordnet sind, erlaubt die Vergabe von Rechten abhängig von Nutzer und Gruppenzugehörigkeit.

Im klassische Unix–Rechtekonzept gehört eine Datei oder ein Verzeichnisses genau:

- einem Benutzer (`user(u)`),
- einer Gruppe (`group(g)`).

Außerdem gibt es noch:

- alle anderen Benutzer (`others(o)`),

die weder besitzender Benutzer noch in der zugeordneten Gruppe sind.

Die Rechte von Benutzer, Gruppe und dem Rest

Für diese Kategorien `user(u)`, `group(g)` und `others(o)` können die Rechte:

- `read(r)`, Leserecht
- `write(w)`, Schreibrecht
- `execute(x)`, Ausführungsrecht

unabhängig vergeben werden[1].

[1] Nicht in Linux Essentials enthalten: ACL (Access Control List) erlauben die Vergabe komplexer Rechtestrukturen ähnlich zu NTFS-Berechtigungen.

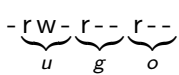
Die Rechte von Benutzer, Gruppe und dem Rest

Für diese Kategorien `user(u)`, `group(g)` und `others(o)` können die Rechte:

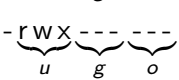
- `read(r)`, Leserecht
- `write(w)`, Schreibrecht
- `execute(x)`, Ausführungsrecht

unabhängig vergeben werden[1]. Beispiele:

→ alle haben alle Rechte



→ Besitzer (u) kann lesen und schreiben, alle anderen nur lesen.

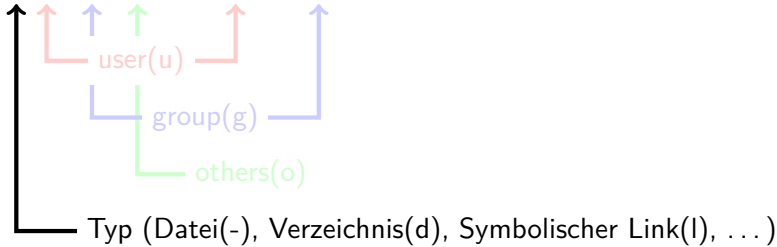


→ Besitzer (u) kann alles, alle anderen nichts.

[1] Nicht in Linux Essentials enthalten: ACL (Access Control List) erlauben die Vergabe komplexer Rechtestrukturen ähnlich zu NTFS-Berechtigungen.

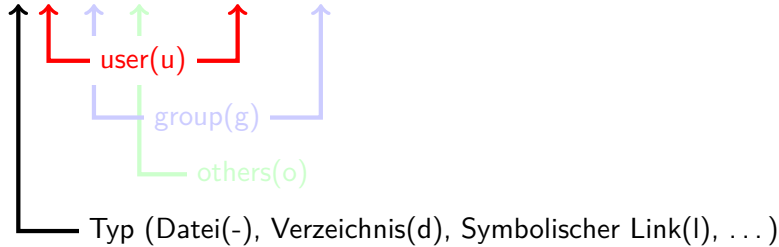
Typ und Rechte anzeigen mit `ls -l`

```
ls -l Example/
total 16
drwxr-xr-x 2 andi adm   4096 Dec 11 13:12 Directory
-rw-r--r-- 1 andi adm     95 Jun  2  2022 File1.txt
-rwxr-xrwx 1 andi andi    71 Sep 12  2020 File2.sh
-rwxr-xr-x 1 andi andi    71 Dec 11 13:03 File4.sh
```



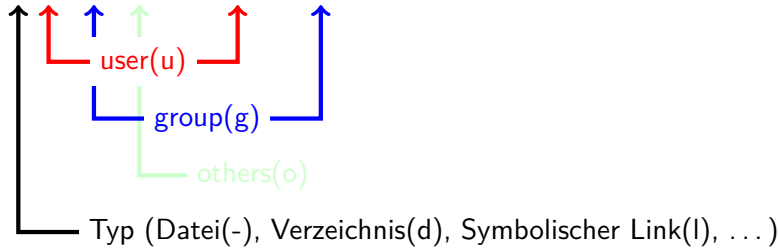
Typ und Rechte anzeigen mit `ls -l`

```
ls -l Example/
total 16
drwxr-xr-x 2 andi adm   4096 Dec 11 13:12 Directory
-rw-r--r-- 1 andi adm     95 Jun  2  2022 File1.txt
-rwxr-xrwx 1 andi andi    71 Sep 12  2020 File2.sh
-rwxr-xr-x 1 andi andi    71 Dec 11 13:03 File4.sh
```



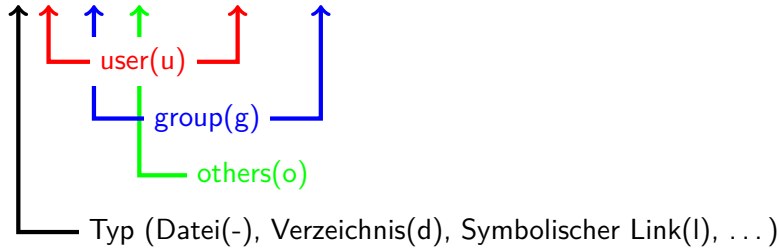
Typ und Rechte anzeigen mit `ls -l`

```
ls -l Example/
total 16
drwxr-xr-x 2 andi adm   4096 Dec 11 13:12 Directory
-rw-r--r-- 1 andi adm     95 Jun  2  2022 File1.txt
-rwxr-xrwx 1 andi andi    71 Sep 12  2020 File2.sh
-rwxr-xr-x 1 andi andi    71 Dec 11 13:03 File4.sh
```



Typ und Rechte anzeigen mit `ls -l`

```
ls -l Example/
total 16
drwxr-xr-x 2 andi adm   4096 Dec 11 13:12 Directory
-rw-r--r-- 1 andi adm     95 Jun  2  2022 File1.txt
-rwxr-xrwx 1 andi andi    71 Sep 12  2020 File2.sh
-rwxr-xr-x 1 andi andi    71 Dec 11 13:03 File4.sh
```



Rechte modifizieren mit chmod

Die Rechte können vom Besitzer verändert werden.
Wir starten mit:

```
ls -l Example/File1.txt
-rw-r--r-- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Der Gruppe und allen anderen („others“) das Schreibrecht geben:

```
chmod go+w Example/File1.txt
ls -l Example/File1.txt
-rw-rw-rw- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

„Others“ soll doch nicht schreiben können:

```
chmod o-w Example/File1.txt
ls -l Example/File1.txt
-rw-rw-r-- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Besitzer, Gruppe und alle anderen nur Schreibrecht geben:

```
chmod ugo=r Example/File1.txt
ls -l Example/File1.txt
-r--r--r-- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Alternativ:

```
chmod a=rw Example/File1.txt
ls -l Example/File1.txt
-rw-rw-rw- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Folgende Syntax wendet die Rechte wie `a=rwx`, aber abzüglich der `umask`[2] an:

```
chmod =rwx Example/File1.txt
ls -l Example/File1.txt
-rwxr-xr-x 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

[2] https://en.wikipedia.org/wiki/Umask

```
ls -l Example/File1.txt
-rwxr-xr-x 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Allen anderen (others) keine Rechte zugestehen:

```
chmod o= Example/File1.txt
ls -l Example/File1.txt
-rwxr-x--- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Rechte für Besitzer, Gruppe und alle anderen direkt definieren:

```
chmod u=rw,g=r,o=rwx Example/File1.txt
ls -l Example/File1.txt
-rw-r--rwx 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Rechte rekursiv modifizieren mit `chmod -R`

Um die Zugriffsrechte aller Dateien eines Verzeichnisses (und alle Unterverzeichnisse sowie deren Dateien) zu modifizieren:

```
ls -l Example/
total 16
drwxr-xr-x 2 andi adm  4096 Dec 11 13:12 Directory
-rw-r--rwx 1 andi adm    95 Jun  2  2022 File1.txt
-rwxr-xrwx 1 andi andi   71 Sep 12  2020 File2.sh
-rwxr-xr-x 1 andi andi   71 Dec 11 13:03 File4.sh
```

```
chmod -R o-rwx Example/
ls -l Example/
total 16
drwxr-x--- 2 andi adm  4096 Dec 11 13:12 Directory
-rw-r----- 1 andi adm    95 Jun  2  2022 File1.txt
-rwxr-x--- 1 andi andi   71 Sep 12  2020 File2.sh
-rwxr-x--- 1 andi andi   71 Dec 11 13:03 File4.sh
```

Ausführungsrechte modifizieren: x und X

Das Ausführungsrecht wird mit X nur gesetzt wenn es sich um ein Verzeichnis handelt oder bereits ein anderes Ausführungsbit gesetzt ist:

```
ls -l Example/
total 16
drwxr-x--- 2 andi adm  4096 Dec 11 13:12 Directory
-rw-r----- 1 andi adm    95 Jun  2  2022 File1.txt
-rwxr-x--- 1 andi andi   71 Sep 12  2020 File2.sh
-rwxr-x--- 1 andi andi   71 Dec 11 13:03 File4.sh
```

```
chmod -R o+rwX Example/
ls -l Example/
total 16
drwxr-xrwx 2 andi adm  4096 Dec 11 13:12 Directory
-rw-r--rw- 1 andi adm    95 Jun  2  2022 File1.txt
-rwxr-xrwx 1 andi andi   71 Sep 12  2020 File2.sh
-rwxr-xrwx 1 andi andi   71 Dec 11 13:03 File4.sh
```

Rechtevergabe in numerischer Syntax

Die Rechte lesen(r), schreiben(w) und ausführen(x) werden jeweils durch ein Bit, also insgesamt 3 Bits beschrieben. Diese 3 Bits können durch eine Oktalzahl-Ziffer $(0, 1, \ldots, 7)$ dargestellt werden:

$$
\begin{array}{lcll}
\mathsf{r\,w\,x} & & & \\
1\,1\,1 & \longleftrightarrow & 2^2 + 2^1 + 2^0 & = 7 \\
1\,1\,0 & \longleftrightarrow & 2^2 + 2^1 & = 6 \\
1\,0\,0 & \longleftrightarrow & 2^2 & = 4 \\
\vdots & & & \vdots
\end{array}
$$

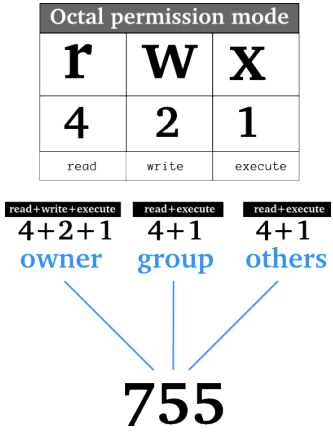
Demnach hat das Ausführungs-Bit den Wert $2^0 = 1$, das Schreib-Bit den Wert $2^1 = 2$ und das Lese-Bit den Wert $2^2 = 4$.

Rechtevergabe in numerischer Syntax

Alle $8 = 2^3$ möglichen Rechtekombinationen lassen sich für Besitzer(u), Gruppe(g) und alle anderen(o) durch jeweils eine Ziffer zwischen $(0, \ldots, 7)$ darstellen:

Oktalzahl	Symbole
0	`---`
1	`--x`
2	`-w-`
$3 = 2 + 1$	`-wx`
4	`r--`
$5 = 4 + 1$	`r-x`
$6 = 4 + 2$	`rw-`
$7 = 4 + 2 + 1$	`rwx`

Häufig für Dateien: 644 $\longleftrightarrow$ `rw-r--r--`
Häufig für Verzeichnisse: 755 $\longleftrightarrow$ `rwxr-xr-x`



Beispiel Datei:

```
ls -l Example/File1.txt
-rw-r--rw- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

```
chmod 600 Example/File1.txt
ls -l Example/File1.txt
-rw------- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

```
chmod 644 Example/File1.txt
ls -l Example/File1.txt
-rw-r--r-- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Beispiel Verzeichnis:

```
ls -ld Example
drwxr-xrwx 3 andi andi 4096 Dec 11 13:03 Example
```

```
chmod 755 Example
ls -ld Example
drwxr-xr-x 3 andi andi 4096 Dec 11 13:03 Example
```

Bedeutung der Rechte für Verzeichnisse

Dateien, Typ: -

- `read(r)`, Leserecht
- `write(w)`, Schreibrecht
- execute(x), Ausführungsrecht

Verzeichnisse, Typ: `d`

- `read(r)`, Namen der Inhalte des Verzeichnisses auflisten.
- `write(w)`, Namen von Inhalten im Verzeichnis anlegen/löschen (in der Praxis `x` ebenfalls erforderlich).
- execute(x), auf zum Namen gehörende Inode-Nummer (d.h. Datei-Metadaten und Datei-Inhalt) zugreifen oder ins Verzeichnis wechseln.

Bedeutung der Rechte für Verzeichnisse

Dateien, Typ: `-`

- `read(r)`, Leserecht
- `write(w)`, Schreibrecht
- execute(x), Ausführungsrecht

Verzeichnisse, Typ: `d`

- `read(r)`, Namen der Inhalte des Verzeichnisses auflisten.
- `write(w)`, Namen von Inhalten im Verzeichnis anlegen/löschen (in der Praxis `x` ebenfalls erforderlich).
- execute(x), auf zum Namen gehörende Inode-Nummer (d.h. Datei-Metadaten und Datei-Inhalt) zugreifen oder ins Verzeichnis wechseln.

Bedeutung der Rechte für Verzeichnisse

Dateien, Typ: -

- `read`(r), Leserecht
- `write`(w), Schreibrecht
- execute(x), Ausführungsrecht

Verzeichnisse, Typ: `d`

- `read`(r), Namen der Inhalte des Verzeichnisses auflisten.
- `write`(w), Namen von Inhalten im Verzeichnis anlegen/löschen (in der Praxis `x` ebenfalls erforderlich).
- execute(x), auf zum Namen gehörende Inode-Nummer (d.h. Datei-Metadaten und Datei-Inhalt) zugreifen oder ins Verzeichnis wechseln.

Bedeutung der Rechte für Verzeichnisse

Dateien, Typ: -

- `read`(r), Leserecht
- `write`(w), Schreibrecht
- execute(x), Ausführungsrecht

Verzeichnisse, Typ: `d`

- `read`(r), Namen der Inhalte des Verzeichnisses auflisten.
- `write`(w), Namen von Inhalten im Verzeichnis anlegen/löschen (in der Praxis `x` ebenfalls erforderlich).
- execute(x), auf zum Namen gehörende Inode-Nummer (d.h. Datei-Metadaten und Datei-Inhalt) zugreifen oder ins Verzeichnis wechseln.

Bedeutung der Rechte für Verzeichnisse

Dateien, Typ: -

- `read`(r), Leserecht
- `write`(w), Schreibrecht
- execute(x), Ausführungsrecht

Verzeichnisse, Typ: `d`

- `read`(r), Namen der Inhalte des Verzeichnisses auflisten.
- `write`(w), Namen von Inhalten im Verzeichnis anlegen/löschen (in der Praxis `x` ebenfalls erforderlich).
- execute(x), auf zum Namen gehörende Inode-Nummer (d.h. Datei-Metadaten und Datei-Inhalt) zugreifen oder ins Verzeichnis wechseln.

Bedeutung der Rechte für Verzeichnisse

Dateien, Typ: `-`

- `read(r)`, Leserecht
- `write(w)`, Schreibrecht
- execute(x), Ausführungsrecht

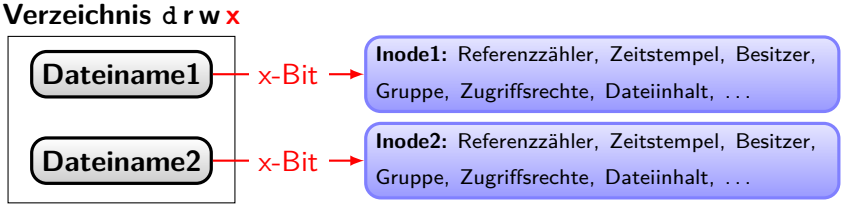
Verzeichnisse, Typ: `d`

- `read(r)`, Namen der Inhalte des Verzeichnisses auflisten.
- `write(w)`, Namen von Inhalten im Verzeichnis anlegen/löschen (in der Praxis `x` ebenfalls erforderlich).
- execute(x), auf zum Namen gehörende Inode-Nummer (d.h. Datei-Metadaten und Datei-Inhalt) zugreifen oder ins Verzeichnis wechseln.

Weitere Typen: `l`: symbolischer Link, `b`: block device, `c`: character device, `p`: (named) pipe, `s`: socket file

Wirkung des execute(x)-Bits bei Verzeichnissen

Erinnerung: Stark vereinfachter Aufbau eines Dateisystem:



Gesetztes execute(x)-Bit erlaubt die zu einem Dateinamen gehörende Inode zu referenzieren, also auf die eigentliche Datei zuzugreifen. Sehr viele Operationen greifen auf die Inode zu (auch wenn sie nicht den eigentlichen Dateiinhalt ändern) und scheitern bei nicht gesetztem execute(x)-Bit[3].

[3] Sehr gute und detaillierte Beschreibung:
`http://wpollock.com/AUnix1/FilePermissions.htm`

Übliche Zugriffsrechte von Verzeichnissen

```
ls -ld /root
drwx------ 9 root root 4096 Dec 10 20:29 /root
```

$\longrightarrow$ Nur root hat alle Rechte

```
ls -ld /etc/
drwxr-xr-x 171 root root 12288 Dec 10 20:30 /etc/
```

$\longrightarrow$ root hat alle Rechte, die anderen dürfen Verzeichnisinhalte ansehen und sich auch im Verzeichnisbaum „bewegen“.

- Home-Verzeichnisse entweder `drwxr-xr-x` oder `drwx------`, je nach Umfeld.
- Oder auch `drwxr-x--x`, wenn auf tiefer liegende Unterverzeichnisse (z.B. `public_html/`) Zugegriff erlaubt werden soll.

Übliche Zugriffsrechte von Verzeichnissen

```
ls -ld /root
drwx------ 9 root root 4096 Dec 10 20:29 /root
```

$\longrightarrow$ Nur root hat alle Rechte

```
ls -ld /etc/
drwxr-xr-x 171 root root 12288 Dec 10 20:30 /etc/
```

$\longrightarrow$ root hat alle Rechte, die anderen dürfen Verzeichnisinhalte ansehen und sich auch im Verzeichnisbaum „bewegen“.

- Home-Verzeichnisse entweder `drwxr-xr-x` oder `drwx------`, je nach Umfeld.
- Oder auch `drwxr-x--x`, wenn auf tiefer liegende Unterverzeichnisse (z.B. `public_html/`) Zugegriff erlaubt werden soll.

Übliche Zugriffsrechte von Verzeichnissen

```
ls -ld /root
drwx------ 9 root root 4096 Dec 10 20:29 /root
```

⟶ Nur root hat alle Rechte

```
ls -ld /etc/
drwxr-xr-x 171 root root 12288 Dec 10 20:30 /etc/
```

⟶ root hat alle Rechte, die anderen dürfen Verzeichnisinhalte ansehen und sich auch im Verzeichnisbaum „bewegen“.

- Home-Verzeichnisse entweder `drwxr-xr-x` oder `drwx------`, je nach Umfeld.
- Oder auch `drwxr-x--x`, wenn auf tiefer liegende Unterverzeichnisse (z.B. `public_html/`) Zugegriff erlaubt werden soll.

Besitzer/Gruppe einer Datei/eines Verzeichnisses ändern

Der Befehl chown (change owner) ändert den Besitzer einer Datei oder eines Verzeichnisses. Dies ist nur mit root-Rechten möglich:

```
#chown guest Example/File1.txt  ## Fehler, sudo oder root-Rechte erforderlich
```

Man kann Besitzer und Gruppe gleichzeitig setzen:

```
chown andi:lpadmin Example/File1.txt
ls -l Example/File1.txt
-rw-r--r-- 1 andi lpadmin 95 Jun  2  2022 Example/File1.txt
```

Oder nur die Gruppe:

```
chown :adm Example/File1.txt
ls -l Example/File1.txt
-rw-r--r-- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Oder Besitzer samt seiner primären Gruppe:

```
chown andi: Example/File1.txt
ls -l Example/File1.txt
-rw-r--r-- 1 andi andi 95 Jun  2  2022 Example/File1.txt
```

Soll nur die Gruppe geändert werden, kann man auch chgrp (change group) verwenden:

```
chgrp adm Example/File1.txt
ls -l Example/File1.txt
-rw-r--r-- 1 andi adm 95 Jun  2  2022 Example/File1.txt
```

Die Befehle chown und chgrp lassen sich wie chmod mit der Option -R rekursiv auf ganze Verzeichnisbäume anwenden.

umask und Standardrechte

Neu angelegte Datei/Verzeichis hat im Standardfall:

Datei 0666 bzw. `rw-rw-rw-`

Verzeichnis 0777 bzw. `rwxrwxrwx`

Die *umask* definiert eine Oktalzahl, die vom Standard subtrahiert wird:

```
umask
rm -rf /tmp/VERZ ; mkdir /tmp/VERZ
touch /tmp/VERZ/DATEI ; ls -dl /tmp/VERZ /tmp/VERZ/DATEI
0022
drwxr-xr-x 2 andi andi 60 Dec 13 21:31 /tmp/VERZ
-rw-r--r-- 1 andi andi  0 Dec 13 21:31 /tmp/VERZ/DATEI
```

```
rm -r /tmp/VERZ ; umask 0027 ; mkdir /tmp/VERZ
touch /tmp/VERZ/DATEI ; ls -dl /tmp/VERZ /tmp/VERZ/DATEI
drwxr-x--- 2 andi andi 60 Dec 13 21:31 /tmp/VERZ
-rw-r----- 1 andi andi  0 Dec 13 21:31 /tmp/VERZ/DATEI
```

Zusammenfassung

Links und Quellen

- Abschnitt 17.7 NDG
- Sehr schöner Überblick:
  `https://de.wikipedia.org/wiki/Unix-Dateirechte`
- Execute(`x`)-Bit bei Verzeichnissen:
  `http://wpollock.com/AUnix1/FilePermissions.htm`
- `man`-Pages diverser Kommandos, insbesondere `chmod`

Aufgaben und Übungen

Typ, Rechte, Besitzer und Gruppe anzeigen mit `ls -l`

1. Was bewirkt die Option –d beim Aufruf von `ls`?
2. Sehen Sie sich Typ, Rechte, Besitzer und Gruppe von verschiedenen Elementen des Dateisystems an:
   - Homeverzeichnis
   - im Homeverzeichnis abgelegte Dateien
   - ihre Konfigurationen (zu finden in meist versteckten Dateien, wie z.B. .config/
   - Systemkonfigurationen unter /etc/
   - System Log-Dateien unter /var/log/
   - Suchen Sie Dateien mit vorgegebenen Rechten, Besitzer und Gruppe[a]. Warum wurden die Rechte entsprechend vergeben?

[a] Beispiel: `find /etc -perm 600 -user root -group root`

Rechte modifizieren mit chmod

1. Legen Sie ein Verzeichnis `Example/` und darin weitere Dateien und Verzeichnisse an.
2. Mit welchen Rechten werden Dateien bzw. Verzeichnisse standardmäßig erzeugt?
3. Ändern Sie die Rechte an einer Datei auf verschiedene Weise:
   - Direkte Zuweisung für Besitzer, Gruppe und alle anderen in einem Befehl:
     - symbolische Notation
     - Oktalnotation
   - Ändern Sie einzelne Rechte ohne Auswirkung auf bestehende.
4. Ändern Sie rekursiv die Rechte von bzw. in `Example/`. Prüfen Sie den Unterschied beim Setzen des execute-Bits durch `x` und `X` nach.
5. Untersuchen Sie die Auswirkung der Rechte eines Verzeichnisses auf erzeugen, löschen, lesen, schreiben und umbenennen enthaltener Dateien.

Besitzer und Gruppe ändern mit chown und chgrp

1. Ändern Sie Besitzer und Gruppe einer Datei auf verschiedene Weise:
   - Besitzer und Gruppe unabhängig mit einem Befehl.
   - Besitzer und Gruppe, dabei soll die Gruppe die primäre Gruppe des Besitzers sein.
   - Nur den Besitzer.
   - Nur die Gruppe, auf zwei verschiedene Arten.
2. Ändern Sie rekursiv Besitzer und Gruppe von bzw. in ihrem `Example/`-Verzeichnis.